(1)

MTI-R91-004

Test Range Tracking Network Processors

Mitchell R. Belzer
Shi B. Chong
Yong M. Cho

Mentor Technologies, Inc.

1992

US Army
White Sands Missile Range

"Views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision unless so designated by other official documentation."

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a. REPORT SECURITY CLASSIFICATION unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT approved for public release; distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Mentor Technologies, Inc. / 6b. OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION U.S. Army White Sands Missile Range
6c. ADDRESS (City, State, and ZIP Code) 12750 Twinbrook Parkway, Suite 101 Rockville, Maryland 20852	7b. ADDRESS (City, State, and ZIP Code) Commanding Officer, STEWS-ID-T U.S. Army White Sands Missile Range New Mexico 88002-5143
8a. NAME OF FUNDING / SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAD07-91-C-0153
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
1865502A	665602		

11. TITLE (Include Security Classification)

Test Range Tracking Network Processors

12. PERSONAL AUTHOR(S)

M. Belzer, Y. Cho, S. Chong

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Technical	FROM 20Jun91 TO 05Mar92	92 March 23	

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	integrated tracking: decentralized estimation: video trackin

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

See back page:

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Foo Lam	22b TELEPHONE (Include Area Code) (505)678-3010 / 22c. OFFICE SYMBOL STEWS-ID-T

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

Abstract

Test Range Tracking Network Processors

Creating accurate tracks of multiple airborne targets from multiple sensors, in real time, can be a computationally demanding process. Our approach is to perform hypothesis testing based upon the traditional method of Maximum Likelihood, but within a distributed filtering environment. This results in a large reduction in the number of floating point computations required to generate the complete set of likelihood function values.

This final report describes results obtained over a 6 month Phase I project. The primary mathematical operation performed by the distributed filter is matrix triangularization. Thus, this research focused on understanding algorithms for performing this operation, as well as their parallelization.

Three methods based on the orthogonal reduction were reviewed. They are the Householder, Givens, and Fast Givens methods. Gaussian elimination seemed to be an attractive alternative in that it is less costly than those based on orthogonal reduction, but this method is not numerically stable and requires pivoting.

An analysis of computational cost was performed for Householder and Fast Givens methods. Although the Householder method is superior to the Fast Givens method for a generally dense matrix factorization, the Fast Givens method well outperforms the Householder in triangularizing the Local Time Update, Local Measurement Update, and Global Measurement Update matrices of our distributed filter. On the other hand, triangularization of the filter's Global Time Update matrix was more efficiently done using the Householder transformation. This is due to the sparse data structure of the matrix.

The concept of downdating and updating was reviewed along with algorithms from LINPACK. While the updating process is no different from a total annihilation process of the newly added observations (appearing as rows of data), downdating is a backwards process which removes the contribution made by the eliminated observations, from the transformed (triangularized) matrix. The cost of downdating was obtained based on the subroutine "SCHDD" from LINPACK.

The "Sameh and Kuck's" scheme and "Greedy" scheme were reviewed as possibilities for parallelization. Both schemes were modified in order to best suit the block upper-triangular nature of the system matrices. Finally, a hardware processing "cell" which performs a plane rotation using the Fast Givens method, was designed. A linear array of cells following Sameh and Kuck's parallel scheme was proposed.

Contents

List of Figures

List of Tables

List of Symbols, Abbreviations, and Acronyms

i	superscripted local system number
j	superscripted vector element number
k	time index, may be subscripted or enclosed in parentheses
q	dimension of the process noise vector
n	dimension of the state vector
m	dimension of the measurement vector
$\surd$	Square root
G_{ij}	Givens Transformation Matrix
M	number of sensors
Q	Orthogonal Matrix
R	Upper Triangular Matrix
w_i	real column vector
DSRIF	Decentralized Square Root Information Filter
GMU	Global Measurement Update
GTU	Global Time Update
LMU	Local Measurement Update
LTU	Local Time Update
MTI	Mentor Technologies, Inc.
WSMR	White Sands Missile Range

Summary

Creating accurate tracks of multiple airborne targets from multiple sensors, in real time, can be a computationally demanding process. Measurements from each sensor must first be correlated with each track. Then, after a correct association is made, the track can be updated to derive a new estimate. A variety of algorithms for performing these processes of "data association" and "track updating" have been described in the literature. Our approach is to perform hypothesis testing based upon the traditional method of Maximum Likelihood, but within a distributed filtering environment. This results in a large reduction in the number of floating point computations required to generate the complete set of likelihood function values.

This final report describes results obtained over a 6 month Phase I project. The primary mathematical operation performed by the distributed filter is matrix triangularization. Thus, this research focused on understanding algorithms for performing this operation, as well as their parallelization.

Three methods based on the orthogonal reduction were reviewed. They are the Householder, Givens, and Fast Givens methods. Gaussian elimination seemed to be an attractive alternative in that it is less costly than those based on orthogonal reduction, but this method is not numerically stable and requires pivoting.

An analysis of computational cost was performed for Householder and Fast Givens methods. Although the Householder method is superior to the Fast Givens method for a generally dense matrix factorization, the Fast Givens method well outperforms the Householder in triangularizing the Local Time Update, Local Measurement Update, and Global Measurement Update matrices of our distributed filter. On the other hand, triangularization of the filter's Global Time Update matrix was more efficiently done using the Householder transformation. This is due to the sparse data structure of the matrix.

The concept of downdating and updating was reviewed along with algorithms from LINPACK. While the updating process is no different from a total annihilation process of the newly added observations (appearing as rows of data), downdating is a backwards process which removes the contribution made by the eliminated observations, from the transformed (triangularized) matrix. The cost of downdating was obtained based on the subroutine "SCHDD" from LINPACK.

The "Sameh and Kuck's" scheme and "Greedy" scheme were reviewed as possibilities for parallelization. Both schemes were modified in order to best suit the block upper-triangular nature of the system matrices. Finally, a hardware processing "cell" which performs a plane rotation using the Fast Givens method, was designed. A linear array of cells following Sameh and Kuck's parallel scheme was proposed.

1. Introduction

Creating tracks or estimates of the position and velocity (and other dynamical states) of multiple targets from a network of multiple sensors, in real time, can be a computationally demanding process. Measurements from each sensor must first be correlated with each track. Then, after a correct association is made, the track can be updated to derive a new estimate. A variety of algorithms for performing these processes of "data association" and "track updating" have been described in the literature. This is an active area of research, whose goal is to produce tracks with ever increasing accuracy. Military applications of the research are improvements in weapon system performance as well as their test and evaluation.

The amount of computation performed in this two step process increases nonlinearly with the number of targets and the number of sensors. The nonlinearity is due to the combinatorial nature of the "data association" process, which is portrayed in Figure 1.-1. In this worst case, when there are t targets and each one of M sensors sees all of the targets, the total number of association hypotheses at each time step could be as high as t^{M+1}. However, if one proceeds track by track, eliminating correctly associated measurements from further consideration, then the number of hypotheses per track would continuously decrease. In this case, the total number of association hypothesis at each time step would be reduced to $t^M + (t-1)^M + \dots + 1^M$.

As an example, when $t=10$ and $M=3$, which corresponds to a typical range scenario for testing an MLRS rocket with 6 submunitions aboard, there are 10,000 possible associations but only 3025 when correct associations are sequentially eliminated. Although this represents a decrease in the number of hypotheses by 69.8%, still the problem is computationally intensive. At data rates of 120 samples per second, 1 hypothesis must be generated and tested every 2.75 microseconds!

After all possible associations are computed (and filtered) locally at each iteration, the resultant set of smoothing coefficients must be sent to the global processor and merged to obtain a set of optimal solutions, each one corresponding to a different association. The globally optimal one stems from the association which admits the maximum value for the likelihood function. The set of optimal solutions may be generated by first deleting an almost upper triangular block of a large matrix (large columns), and adding a new one corresponding to a different association. Then, Householder transformations or Givens rotations may be applied to the matrix, putting it into upper triangular form. The latter two steps are repeated many times until all of the hypotheses have been tested. Thus, the focus of this research is the development of specific algorithms for performing the transformations and/or rotations as fast as possible.

One particularly interesting idea is to first "downdate" the old association under test directly from the **upper triangular matrix**, and then "update" it with the new association. Thus, working on a relatively full and large matrix can be avoided and much computation can be saved.

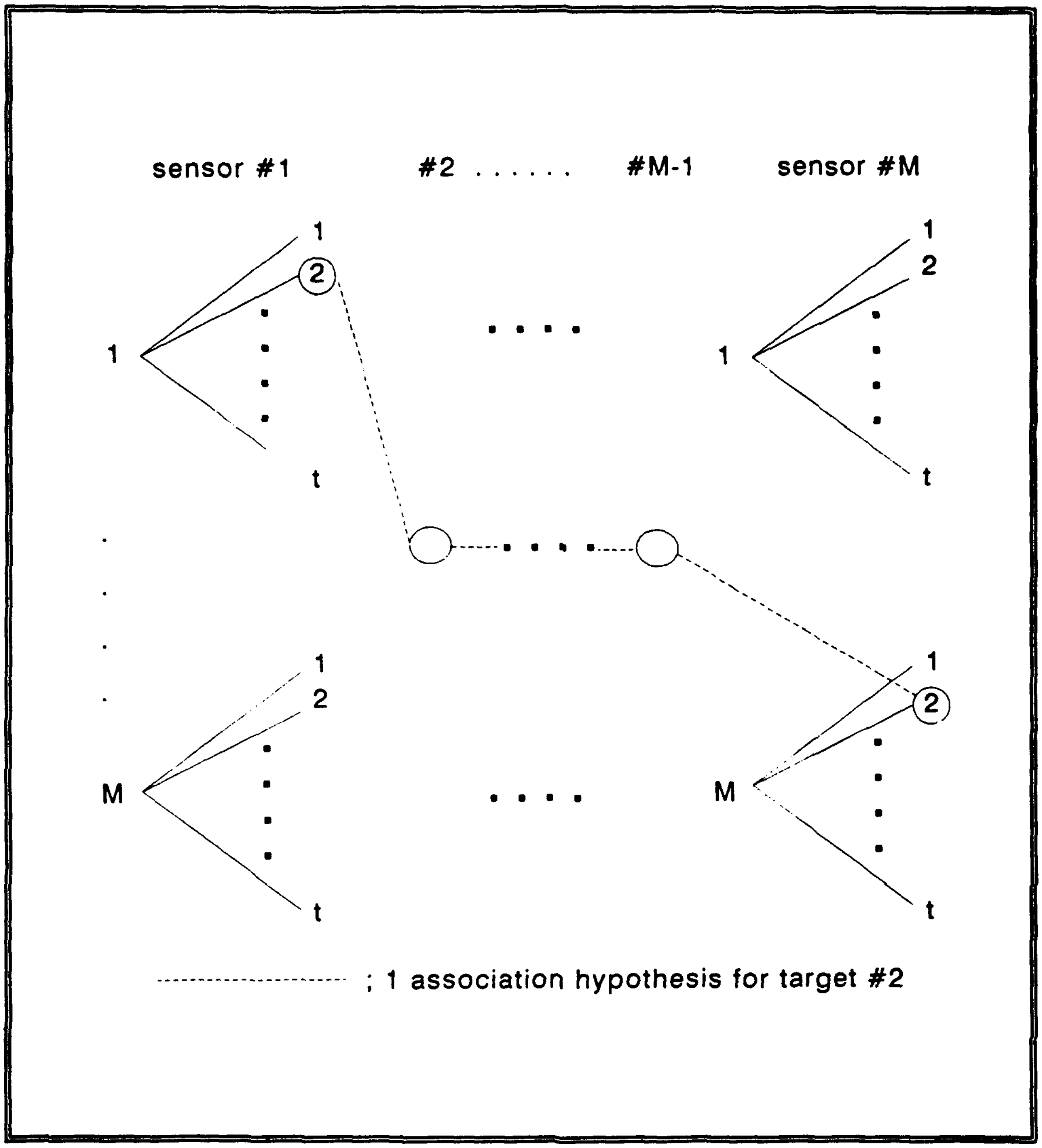


Figure 1.-1: Pictoral of a Single Hypothesis for Data Association

2. Work Carried Out/Results Obtained

2.1 The Decentralized Square Root Information Filter

The Decentralized Square Root Information Filter (DSRIF) was introduced in [1,2]. It is a distributed solution to the constrained linear least squares estimation problem, and can be used to update and extrapolate tracks. For 1 target, the DSRIF admits the following matrix structure shown in Figure 2.1-1. As seen, it includes (i)local time update, (ii)local measurement update, (iii)global time update and (iv)global measurement update matrices which are block upper-triangular! In Figure 2.1-1,

q = dimension of the process noise vector (typically 3)
n = dimension of the state vector (typically 6 or 9)
m = dimension of the measurement vector (assumed to be the same for all sensors, actually m equals 2 or 3)
M = number of sensors, and we assume that each sensor sees **all** of the targets

For multiple targets, we have the following structures where t is equal to the total number of targets being tracked by the network

(i) t local time updates which can all be done in parallel
(ii) t^2 completely independent local measurement updates which can all be done in parallel i.e., there is no recursion
(iii) t global time updates which can all be done in parallel
(iv) t^{M+1} global measurement updates, each corresponding to a different set of associations of measurements with target tracks. t^{M+1} is equal to the maximum number of hypotheses.

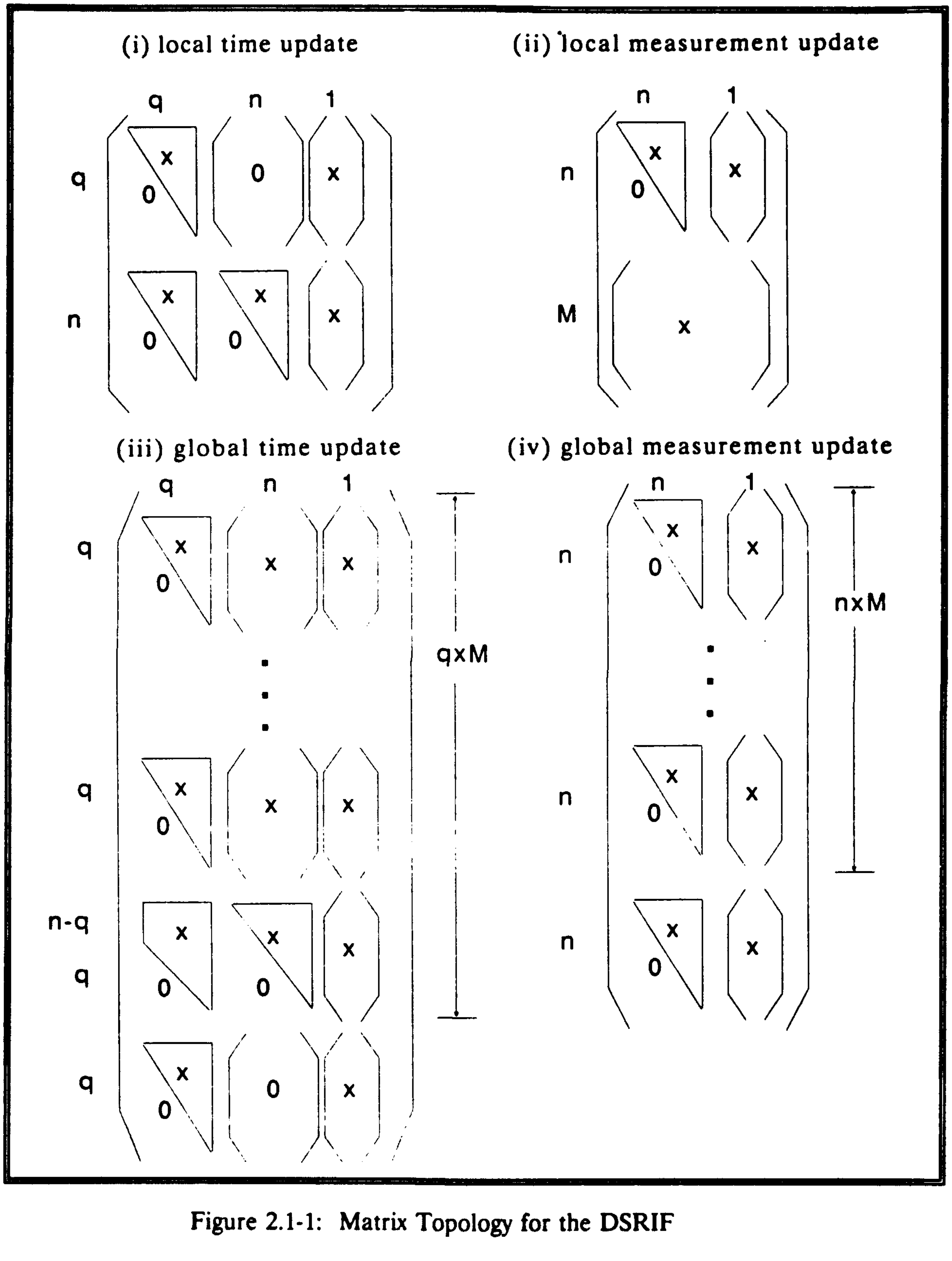


Figure 2.1-1: Matrix Topology for the DSRIF

2.2 Survey of Microprocessors

Microprocessors and math coprocessors that are commercially available in multiprocessing board sets are listed in Tables 2.2-1 and 2.2-2.

Table 2.2-1: Math Coprocessors

Manufacturer	Processor	Clock Speed (MHz)	MIPS	Benchmark Speed (MFLOPS)	Price $
Cyrix	83S87	20 MHz			556
	83D87	33 MHz			994
	EMC87	33 MHz			994
Intel	8087	5 MHz			142
	80287XL	12.5 MHz			326
	387SX	20 MHz			550
	387DX	33 MHz			994
	i486	33 MHz			667
Motorola	68881	20 MHz			68
	68882	40 MHz			218
Weitek	3167	33 MHz			995
	4167	33 MHz			1,295

Our survey revealed that currently, several hundred Mflop systems are available for under 10 thousand dollars. However, a 300 Mflop multiboard set can perform only 825 floating point operations in 2.75 microseconds (continuing the example from section 1.). This is approximately 1 tenth of the total number of floating point computations needed to time update and measurement update the DSRIF. Thus, there is motivation for continuing this research in the pages that follow.

Table 2.2-2: General Purpose Microprocessors

Manufacturer	Processor	Clock Speed (MHz)	MIPS	Benchmark Speed (MFLOPS)	Price $
Advance Micro Devices Inc.	AM29050	40 MHz	32	80	410
Cypress/Ross Technology Inc.	CY7C601	40 MHz	29		805
Fujitsu Micro-Electronics	MB86903	40 MHz	29	5	350
Integrated Device Technology Inc.	79R3000A	40 MHz	33	11	275
Intel Corp.	i860	40 MHz	57.5	10	567
Motorola Inc.	88100	33 MHz	28	5.4	150
Performance Semiconductor	PIMM	40 MHz	33		1,300
VLSI Technology Inc.	VL86C020	25 MHz			100

2.3 Matrix Upper Triangularization

Householder, Givens and Fast Givens transformation techniques were investigated during the Phase I work. Analytical expressions for their computational cost in terms of dimensional parameters were derived. These expressions are useful in deciding which technique is most efficient.

2.3.1 Description of the Householder Method

Following [3], a general matrix, $A \in \mathbb{R}^{m\times n}$, can be upper triangularized.

$$A = QR$$

where Q is an orthogonal matrix, R is an upper triangular matrix with

$$R = P_k P_{k-1} \dots P_2 P_1 A,$$

where $k=n$ for $m>n$ or $k=m-1$ for $m\leq n$, and $P_i \in \mathbb{R}^{m\times m}$ is a Householder transformation matrix.

$$Q = (P_k P_{k-1} \dots P_2 P_1)^{-1},$$

$$P_i = I - w_i w_i^{tr},$$

where w_i is a real column vector and

$$w_i^{tr} w_i = 2.$$

As a first step to triangularization, we annihilate all elements below the main diagonal in the first column of A. This is shown as follows:

Let

$$\begin{aligned} A_1 &= P_1 A \\ &= (I - w_1 w_1^{tr})A \\ &= A - w_1 w_1^{tr} A \\ &= A - w_1 w_1^{tr}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \end{aligned}$$

where $\mathbf{a}_j$ is the jth column of A.

Let

$$u_1^{tr} = [a_{11}-s_1, a_{21}, a_{31}, \dots , a_{m1}]$$

$$w_1 = \mu_1 u_1,$$

where

$$s_1 = \pm(a_1^{tr} a_1)^{½} = \pm(\text{Euclidian Norm of } \mathbf{a1}),$$

$$\tau_1 = (s_1^2 - a_{11}s_1)^{-1},$$

$$\mu_1 = \tau_1^{½},$$

and the sign of s_1 is chosen to be opposite to that of a_{11} for numerical stability.

$$w_1^{tr} a_1 = \mu_1[(a_{11} - s_1)a_{11} + \sum_{i=2}^{m} a^2_{i1}]$$

$$= \mu_1(s_1^2 - a_{11}s_1)$$

$$= \mu_1/\tau_1$$

$$= 1/\mu_1$$

$$a_{11} - w_1 w_1^{tr} a_1 = a_{11} - \mu_1(a_{11} - s_1)/\mu_1 = s_1$$

$$a_{r1} - w_r w_1^{tr} a_1 = a_{r1} - \mu_1 a_{r1}/\mu_1 = 0 \quad \text{for } r = 2,\dots,m$$

where w_r is the rth element of w_1.

Therefore,

$$A_1 = P_1 A = \begin{bmatrix} s_1 & a'_{12} & \cdots & a'_{1n} \\ 0 & a'_{22} & & \\ \cdot & & & \\ \cdot & & & \\ 0 & a'_{m2} & \cdots & a'_{mn} \end{bmatrix}$$

To annihilate all elements below the main diagonal in the second column of A_1, the previous procedure is repeated with $w_2 = \mu_2 u_2$

where,

$$u_2^{tr} = [0, a'_{22} - s_2, a'_{32}, a'_{42}, \dots, a'_{m2}],$$

and s_2 is the Euclidian norm of the vector which results from the second column of A1, but exclusive of its elements above the main diagonal. The sign of s_2 is taken as opposite to that of a'_{22}.

Continuing, we have

$$A_2 = P_2A_1 = (I - w_2w_2^{tr})A_1.$$

A_2 retains the zero elements in the first column and P_2 introduces new zeros in the last m-2 positions of the second column.

To annihilate all elements below the main diagonal in the third column of A_2, the previous procedure is again repeated, but with

$$u_3^{tr} = [0, 0, a''_{33} - s_3, a''_{43}, \dots, a''_{m3}]$$

and s_3 is the Euclidian norm of the vector which results from the third column of A_2, but exclusive of its elements above the main diagonal. The sign of s_3 is taken as opposite to that of a''_{33}. The a''_{33} is the element at the third row and third column position in the matrix A_2.

We continue in this way to zero elements below the main diagonal in column by column order by applying Householder matrices. Thus the resultant matrix R is an upper triangular matrix,

$$R = P_k \cdots P_1A$$

Now, go back to the stage of obtaining A_1.

$$\begin{aligned} A_1 &= P_1A = (I - w_1w_1^{tr})A \\ &= A - w_1w_1^{tr}A \\ &= A - w_1(w_1^{tr}a_1, w_1^{tr}a_2, \dots, w_1^{tr}a_n) \end{aligned}$$

So the j^{th} column of A_1 is

$$a_j - w_1^{tr}a_jw_1 = a_j - \tau_1u_1^{tr}a_ju_1$$

The reduction of an m x n matrix for m > n, to upper triangular form using Householder transformations is summarized by the following vector pseudocode, in Figure 2.3.1-1.

```
For k = 1 to n

    $s_k = -\mathrm{sgn}(a_{kk}) \left( \sum_{i=k}^{m} a^2_{ik} \right)^{1/2}$,

    $\tau_k = (s_k^2 - s_k a_{kk})^{-1}$

    $\mathbf{u}_k^{tr} = (0, \dots, 0, a_{kk}-s_k, a_{k+1,k}, \dots, a_{mk})$

    $a_{kk} = s_k$

      For j = k+1 to m

        $\alpha_j = \tau_k \mathbf{u}_k^{tr} \mathbf{a}_j$
        $\mathbf{a}_j = \mathbf{a}_j - \alpha_j \mathbf{u}_k$

      End
End
```

Figure 2.3.1-1: Column-oriented Householder reduction

Although column oriented Householder reductions was discussed, the method can also be applied in a row oriented format. This alternative has no advantage over the column oriented format, and therefore was not further investigated.

2.3.2 Computational Cost for Householder Reduction

A more formalized Householder algorithm from [4] is shown below in Figure 2.3.2-1. The algorithm factors an m x n matrix A, overwriting the coefficient matrix with both R and the vectors characterizing each Householder matrix. During the kth step, we annihilate m - k elements in the coefficient matrix using a Householder matrix whose corresponding vector **w** has m - k + 1 non-zero components. Since the non-zero components of **w** do not fit within the space created by the annihilated coefficients, we store the diagonal of R in a separate array **d** to make room for the first non-zero component of **w**. After executing the following algorithm, the diagonal of R is stored in **d**, the remaining elements above R's diagonal are stored above the diagonal in A, and the vectors **w** related to each Householder matrix are stored on and beneath the diagonal of A.

```
k ← 0
for l = 1 to n                          ;column 1 to n
    k ← k+1,  if k = m  then  $d_l ← a_{kl}$  and  exit l loop

    $s = (\sum_{i=k}^{m} a_{il}^2)^{1/2}$

    if s = 0  then  $d_l ← 0$  and  go to next l
    $t ← a_{kl}$,   $r ← 1/[s(s+|t|)]^{1/2}$,   if t < 0  then  s ← -s.
    $d_l ← -s$,   $a_{kk} ← r(t+s)$
    $a_{ik} ← ra_{il}$   for i = k+1 to m

    for j = l+1 to n
        t ← 0
        $t ← t+a_{ik}a_{ij}$   for i = k to m
        $a_{ij} ← a_{ij}-ta_{ik}$   for i = k to m
    next j

next l
```

Figure 2.3.2-1: Householder Reduction Algorithm

The above Householder algorithm annihilates elements column by column in order as shown below for a (6 x 5) matrix as an example.

$$\begin{array}{ccccc l}
x & x & x & x & x & \text{1st row} \\
1 & x & x & x & x & \text{2nd row} \\
1 & 2 & x & x & x & \text{3rd row} \\
1 & 2 & 3 & x & x & \text{4th row} \\
1 & 2 & 3 & 4 & x & \text{5th row} \\
1 & 2 & 3 & 4 & 5 & \text{6th row}
\end{array}$$

The integer number indicates the steps of annihilation. Here annihilation means the value of an element becoming zero.

Costs of the algorithm in Figure 2.3.2-1 is shown in Tables 2.3.2-1 to 2.3.2-6 for $m<n$ and Tables 2.3.2-1 to 2.3.2-9 for $m>n$. We assume that the overhead costs due to communication between registers, initialization of memory, and transfer is negligible compared with the cost of arithmetic operations.

Given: A = [m x n] dense matrix, $m < n$

Table 2.3.2-1. Annihilation of 1st column

All the elements below the main diagonal in the first column become zero.	
$l = 1, k = 1$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{11}^2 + a_{21}^2 + \ldots + a_{ml}^2)^{1/2}$	m x, (m-1) +, 1 √
$t = a_{11}$	
$r = 1/[s(s+\lvert t\rvert)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
1st column:	
$a_{11} = r[t + (\mathrm{sgn}(t))s]$	; "sgn" is sign function -- returns the sign of argument.
$a_{21} = r a_{21}$	
$a_{31} = r a_{31}$	
.	
.	
$a_{ml} = r a_{ml}$	(m+1) x, 1 +, 1 sign
2nd column: $j = l + 1 = 2$	
$t = a_{11}a_{12} + a_{21}a_{22} + a_{31}a_{32} + \ldots + a_{ml}a_{m2}$	m x, (m-1) +
$a_{12} = a_{12} - t a_{11}$	
$a_{22} = a_{22} - t a_{21}$	
$a_{32} = a_{32} - t a_{31}$	
.	
.	
$a_{m2} = a_{m2} - t a_{ml}$	m x, m -
3rd column: $j = l + 1 = 3$	
$t = a_{11}a_{13} + a_{21}a_{23} + a_{31}a_{33} + \ldots + a_{ml}a_{m3}$	m x, (m-1) +
$a_{13} = a_{13} - t a_{11}$	
$a_{23} = a_{23} - t a_{21}$	
$a_{33} = a_{33} - t a_{31}$	
.	
.	
$a_{m3} = a_{m3} - t a_{ml}$	m x, m -
4th column: $j = l + 1 = 4$	
$t = a_{11}a_{14} + a_{21}a_{24} + a_{31}a_{34} + \ldots + a_{ml}a_{m4}$	m x, (m-1) +
$a_{14} = a_{14} - t a_{11}$	
$a_{24} = a_{24} - t a_{21}$	
$a_{34} = a_{34} - t a_{31}$	
.	
.	
$a_{m4} = a_{m4} - t a_{ml}$	m x, m -
.	
.	
nth column: $j = l + 1 = n$	
$t = a_{11}a_{1n} + a_{21}a_{2n} + a_{31}a_{3n} + \ldots + a_{ml}a_{mn}$	m x, (m-1) +
$a_{1n} = a_{1n} - t a_{11}$	
$a_{2n} = a_{2n} - t a_{21}$	
$a_{3n} = a_{3n} - t a_{31}$	
.	
$a_{mn} = a_{mn} - t a_{ml}$	m x, m -

Table 2.3.2-2. Annihilation of 2nd column

All the elements below the main diagonal in the second column become zero.	
$l = 2,\ k = 2$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{22}^2 + a_{32}^2 + \ldots + a_{m2}^2)^{1/2}$	(m-1) x, (m-2) +, 1 √
$t = a_{22}$	
$r = 1/[s(s+\lvert t\rvert)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
2nd column:	
$a_{22} = r[t + (\mathrm{sgn}(t))s]$	
$a_{32} = r a_{32}$	
$a_{42} = r a_{42}$	
.	
.	
$a_{m2} = r a_{m2}$	m x, 1 +, 1 sign
3rd column: $j = l + 1 = 3$	
$t = a_{22}a_{23} + a_{32}a_{33} + a_{42}a_{43} + \ldots + a_{m2}a_{m3}$	(m-1) x, (m-2) +
$a_{23} = a_{23} - t a_{22}$	
$a_{33} = a_{33} - t a_{32}$	
$a_{43} = a_{43} - t a_{42}$	
.	
.	
$a_{m3} = a_{m3} - t a_{m2}$	(m-1) x, (m-1) -
4th column: $j = l + 1 = 4$	
$t = a_{22}a_{24} + a_{32}a_{34} + a_{42}a_{44} + \ldots + a_{m2}a_{m4}$	(m-1) x, (m-2) +
$a_{24} = a_{24} - t a_{22}$	
$a_{34} = a_{34} - t a_{32}$	
$a_{44} = a_{44} - t a_{42}$	
.	
.	
$a_{m4} = a_{m4} - t a_{m2}$	(m-1) x, (m-1) -
.	
.	
nth column: $j = l + 1 = n$	
$t = a_{22}a_{2n} + a_{32}a_{3n} + a_{42}a_{4n} + \ldots + a_{m2}a_{mn}$	(m-1) x, (m-2) +
$a_{2n} = a_{2n} - t a_{22}$	
$a_{3n} = a_{3n} - t a_{32}$	
$a_{4n} = a_{4n} - t a_{42}$	
.	
.	
$a_{mn} = a_{mn} - t a_{m2}$	(m-1) x, (m-1) -

Table 2.3.2-3. Annihilation of 3rd column

All the elements below the main diagonal in the third column become zero.	
$l = 3, k = 3$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{33}^2 + a_{43}^2 + \ldots + a_{m3}^2)^{1/2}$	(m-2) x, (m-3) +, 1 √
$t = a_{33}$	
$r = 1/[s(s+\|t\|)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
3rd column: $j=l+1=3$	
$a_{33} = r[t + (\mathrm{sgn}(t))s]$	
$a_{43} = r a_{43}$	
$a_{53} = r a_{53}$	
.	
.	
$a_{m3} = r a_{m3}$	(m-1) x, 1 +, 1 sign
4th column: $j=l+1=4$	
$t = a_{33}a_{34} + a_{43}a_{44} + a_{53}a_{54} + \ldots + a_{m3}a_{m4}$	(m-2) x, (m-3) +
$a_{34} = a_{34} - t a_{33}$	
$a_{44} = a_{44} - t a_{43}$	
$a_{54} = a_{54} - t a_{53}$	
.	
.	
$a_{m4} = a_{m4} - t a_{m3}$	(m-2) x, (m-2) -
5th column: $j=l+1=5$	
$t = a_{33}a_{35} + a_{43}a_{45} + a_{53}a_{55} + \ldots + a_{m3}a_{m5}$	(m-2) x, (m-3) +
$a_{35} = a_{35} - t a_{33}$	
$a_{45} = a_{45} - t a_{43}$	
$a_{55} = a_{55} - t a_{53}$	
.	
.	
$a_{m5} = a_{m5} - t a_{m3}$	(m-2) x, (m-2) -
.	
.	
nth column: $j=l+1=n$	
$t = a_{33}a_{3n} + a_{43}a_{4n} + a_{53}a_{5n} + \ldots + a_{m3}a_{mn}$	(m-2) x, (m-3) +
$a_{3n} = a_{3n} - t a_{33}$	
$a_{4n} = a_{4n} - t a_{43}$	
$a_{5n} = a_{5n} - t a_{53}$	
.	
.	
$a_{mn} = a_{mn} - t a_{m3}$	(m-2) x, (m-2) -

Successive columns are processed in a similar manner.

Table 2.3.2-4. Annihilation of m-3 column

All the elements below the main diagonal in the m-3 column become zero.	
$l = m\text{-}3,\ k = m\text{-}3$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{m-3,m-3}^2 + a_{m-2,m-3}^2 + .. + a_{m,m-3}^2)^{1/2}$	4 x, 3 +, 1 √
$t = a_{m-3,m-3}$	
$r = 1/[s(s+\lvert t \rvert)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
m-3 column: $j = l+1 = m\text{-}3$	
$a_{m-3,m-3} = r[t + (\mathrm{sgn}(t))s]$	
$a_{m-2,m-3} = r a_{m-2,m-3}$	
$a_{m-1,m-3} = r a_{m-1,m-3}$	
$a_{m,m-3} = r a_{m,m-3}$	5 x, 1 +, 1 sign
m-2 column: $j = l+1 = m\text{-}2$	
$t = a_{m-3,m-3}a_{m-3,m-2} + a_{m-2,m-3}a_{m-2,m-2}$	
$+ a_{m-1,m-3}a_{m-1,m-2} + a_{m,m-3}a_{m,m-2}$	4 x, 3 +
$a_{m-3,m-2} = a_{m-3,m-2} - t a_{m-3,m-3}$	
$a_{m-2,m-2} = a_{m-2,m-2} - t a_{m-2,m-3}$	
$a_{m-1,m-2} = a_{m-1,m-2} - t a_{m-1,m-3}$	
$a_{m,m-2} = a_{m,m-2} - t a_{m,m-3}$	4 x, 4 -
m-1 column: $j = l+1 = m\text{-}1$	
$t = a_{m-3,m-3}a_{m-3,m-1} + a_{m-2,m-3}a_{m-2,m-1}$	
$+ a_{m-1,m-3}a_{m-1,m-1} + a_{m,m-3}a_{m,m-1}$	4 x, 3 +
$a_{m-3,m-1} = a_{m-3,m-1} - t a_{m-3,m-3}$	
$a_{m-2,m-1} = a_{m-2,m-1} - t a_{m-2,m-3}$	
$a_{m-1,m-1} = a_{m-1,m-1} - t a_{m-1,m-3}$	
$a_{m,m-1} = a_{m,m-1} - t a_{m,m-3}$	4 x, 4 -
.	
.	
nth column: $j = l+1 = n$	
$t = a_{m-3,m-3}a_{m-3,n} + a_{m-2,m-3}a_{m-2,n}$	
$+ a_{m-1,m-3}a_{m-1,n} + a_{m,m-3}a_{m,n}$	4 x, 3 +
$a_{m-3,n} = a_{m-3,n} - t a_{m-3,m-3}$	
$a_{m-2,n} = a_{m-2,n} - t a_{m-2,m-3}$	
$a_{m-1,n} = a_{m-1,n} - t a_{m-1,m-3}$	
$a_{m,n} = a_{m,n} - t a_{m,m-3}$	4 x, 4 -

Table 2.3.2-5. Annihilation of m-2 column

All the elements below the main diagonal in the m-2 column become zero.	
$l = m\text{-}2,\ k = m\text{-}2$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{m-2,m-2}^2 + a_{m-1,m-2}^2 + a_{m,m-2}^2)^{1/2}$	3 x, 2 +, 1 $\surd$
$t = a_{m-2,m-2}$	
$r = 1/[s(s+\lvert t\rvert)]^{1/2}$	1 x, 1 +, 1 $\surd$, 1 +, 1 sign (abs)
m-2 column: $j = l + 1 = m\text{-}2$	
$a_{m-2,m-2} = r[t + (\text{sgn}(t))s]$	
$a_{m-1,m-2} = r\,a_{m-1,m-2}$	
$a_{m,m-2} = r\,a_{m,m-2}$	4 x, 1 +, 1 sign
m-1 column: $j = l + 1 = m\text{-}1$	
$t = a_{m-2,m-2}a_{m-2,m-1} + a_{m-1,m-2}a_{m-1,m-1}$	
$+ a_{m,m-2}a_{m,m-1}$	3 x, 2 +
$a_{m-2,m-1} = a_{m-2,m-1} - t\,a_{m-2,m-2}$	
$a_{m-1,m-1} = a_{m-1,m-1} - t\,a_{m-1,m-2}$	
$a_{m,m-1} = a_{m,m-1} - t\,a_{m,m-2}$	
$a_{m,m-1} = a_{m,m-1} - t\,a_{m,m-2}$	3 x, 3 -
.	
.	
nth column: $j = l + 1 = n$	
$t = a_{m-2,m-2}a_{m-2,n} + a_{m-1,m-2}a_{m-1,n}$	
$+ a_{m,m-2}a_{m,n}$	3 x, 2 +
$a_{m-2,n} = a_{m-2,n} - t\,a_{m-2,m-2}$	
$a_{m-1,n} = a_{m-1,n} - t\,a_{m-1,m-2}$	
$a_{m,n} = a_{m,n} - t\,a_{m,m-2}$	3 x, 3 -

Table 2.3.2-6. Annihilation of m-1 column

All the elements below the main diagonal in the m-1 column become zero.	
$l = m-1,\ k = m-1$	
$s = (\sum_{i=k}^{m} a_{ij}^2)^{1/2} = (a_{m-1,m-1}^2 + a_{m,m-1}^2)^{1/2}$	2 x, 1 +, 1 √
$t = a_{m-1,m-1}$	
$r = 1/[s(s+\|t\|)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
m-1 column: $j = l+1 = m-1$	
$a_{m-1,m-1} = r[t + (\mathrm{sgn}(t))s]$	
$a_{m,m-1} = r \cdot a_{m,m-1}$	3 x, 1 +, 1 sign
.	
.	
nth column: $j = l+1 = n$	
$t = a_{m-1,m-1}a_{m-1,n} + a_{m,m-1}a_{m,n}$	2 x, 1 +
$a_{m-1,n} = a_{m-1,n} - t \cdot a_{m-1,m-1}$	
$a_{m,n} = a_{m,n} - t \cdot a_{m,m-1}$	2 x, 2 -

The upper triangularization of an m x n matrix for m<n is complete at this step! However, the upper triangularization is not done if m is greater than n. Therefore, a few more steps of annihilation are presented for a matrix in which m is greater than n.

Table 2.3.2-7. Annihilation of n-2 column

All the elements below the main diagonal in the n-2 column become zero.	
$l = n-2,\ k = n-2$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{n-2,n-2}^2 + a_{n-1,n-2}^2 + .. + a_{m,n-2}^2)^{1/2}$	(m-n+3) x, (m-n+2) +, 1 √
$t = a_{n-2,n-2}$	.
$r = 1/[s(s+\lvert t\rvert)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
n-2 column:	
$a_{n-2,n-2} = r[t + (\mathrm{sgn}(t))s]$	
$a_{n-1,n-2} = r a_{n-1,n-2}$	
$a_{n,n-2} = r a_{n,n-2}$	
.	
.	
$a_{m,n-2} = r a_{m,n-2}$	(m-n+4) x, 1 +, 1 sign
n-1 column: $j = l+1 = n-1$	
$t = a_{n-2,n-2}a_{n-2,n-1} + a_{n-1,n-2}a_{n-1,n-1}$	
$+ \ldots + a_{m,n-2}a_{m,n-1}$	(m-n+3) x, (m-n+2) +
$a_{n-2,n-1} = a_{n-2,n-1} - t a_{n-2,n-2}$	
$a_{n-1,n-1} = a_{n-1,n-1} - t a_{n-1,n-2}$	
.	
.	
$a_{m,n-1} = a_{m,n-1} - t a_{m,n-2}$	(m-n+3) x, (m-n+3) -
nth column: $j = l+1 = n$	
$t = a_{n-2,n-2}a_{n-2,n} + a_{n-1,n-2}a_{n-1,n}$	
$+ \ldots + a_{m,n-2}a_{m,n}$	(m-n+3) x, (m-n+2) +
$a_{n-2,n} = a_{n-2,n} - t a_{n-2,n-2}$	
$a_{n-1,n} = a_{n-1,n} - t a_{n-1,n-2}$	
.	
.	
$a_{m,n} = a_{m,n} - t a_{m,n-2}$	(m-n+3) x, (m-n+3) -

Table 2.3.2-8. Annihilation of n-1 column

All the elements below the main diagonal in the n-1 column become zero.	
$l = n-1,\ k = n-1$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{n-1,n-1}^2 + a_{n,n-1}^2 + .. + a_{m,n-1}^2)^{1/2}$	(m-n+2) x, (m-n+1) +, 1 √
$t = a_{n-1,n-1}$	
$r = 1/[s(s+\|t\|)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
n-1 column:	
$a_{n-1,n-1} = r[t + (sgn(t))s]$	
$a_{n,n-1} = r a_{n,n-1}$	
.	
.	
$a_{m,n-1} = r a_{m,n-1}$	(m-n+3) x, 1 +, 1 sign
nth column: $j = l+1 = n$	
$t = a_{n-1,n-1}a_{n-1,n} + a_{n,n-1}a_{n,n} + ... + a_{m,n-1}a_{m,n}$	(m-n+2) x, (m-n+1) +
$a_{n-1,n} = a_{n-1,n} - t a_{n-1,n-1}$	
$a_{n,n} = a_{n,n} - t a_{n,n-1}$	
.	
.	
$a_{m,n} = a_{m,n} - t a_{m,n-1}$	(m-n+2) x, (m-n+2) -

Table 2.3.2-9. Annihilation of nth column

All the elements below the main diagonal in the nth column become zero.	
$l = n,\ k = n$	
$s = (\sum_{i=k}^{m} a_{il}^2)^{1/2} = (a_{n,n}^2 + a_{n+1,n}^2 + .. + a_{m,n}^2)^{1/2}$	(m-n+1) x, (m-n) +, 1 √
$t = a_{n,n}$	
$r = 1/[s(s+\|t\|)]^{1/2}$	1 x, 1 +, 1 √, 1 ÷, 1 sign (abs)
nth column:	
$a_{n,n} = r[t + (sgn(t))s]$	
$a_{n+1,n} = r a_{n-1,n}$	
.	
.	
$a_{m,n} = r a_{m,n}$	(m-n+2) x, 1 +, 1 sign

The upper triangularization of an m x n matrix for m>n is complete at this step!

Now we collect all the cost terms along with the arithmetic operations from the foregoing procedures (Tables 2.3.2-1 to 2.3.2-6) and summarize them in Table 2.3.2-10 for the case $m<n$.

Combining the costs of arithmetic operations, we have the total cost (Table 2.3.2-11) for upper triangularization of an m x n matrix ($m<n$). Here, we assume that the unit cost for subtraction is the same as that for addition, and the unit cost for division is the same as that for multiplication.

Again, collecting all the cost terms from Tables 2.3.2-1 to 2.3.2-9 and summarizing them, Table 2.3.2-12 is a summary for the case $m>n$.

Combining the costs of arithmetic operations, we have the total cost (Table 2.3.2-13) for upper triangularization of an m x n matrix ($m>n$).

Table 2.3.2-10. Summary of Cost for Upper Triangularization of an m x n dense matrix, m<n.

Oper.	l = 1	l = 2	l = 3 ...	l = m-3	l = m-2	l = m-1	subtotal
x	m	m-1	m-2	4	3	2	(m+2)(m-1)/2
+	m-1	m-2	m-3	3	2	1	m(m-1)/2
√	1	1	1	1	1	1	m-1
x	1	1	1	1	1	1	m-1
+	1	1	1	1	1	1	m-1
√	1	1	1	1	1	1	m-1
÷	1	1	1	1	1	1	m-1
sign	1	1	1	1	1	1	m-1
x	m+1	m	m-1	5	4	3	(m+4)(m-1)/2
+	1	1	1	1	1	1	m-1
sign	1	1	1	1	1	1	m-1
x	2m(n-1)	2(m-1)(n-2)	2(m-2)(n-3)	2(4)(n-m+3)	2(3)(n-m+2)	2(2)(n-m+1)	$(-1/3)m^3+(n-1)m^2+(n+4/3)m-2n$
+	(m-1)(n-1)	(m-2)(n-2)	(m-3)(n-3)	3(n-m+3)	2(n-m+2)	1(n-m+1)	Σ1
-	m(n-1)	(m-1)(n-2)	(m-2)(n-3)	4(n-m+3)	3(n-m+2)	2(n-m+1)	Σ2

$\Sigma1 + \Sigma2 = (2m-1)(n-1) + (2m-3)(n-2) + (2m-5)(n-3) + \ldots + 5(n-m-2) + 3(n-m+1) = (-1/3)m^3+(n-1/2)m^2+(5/6)m-n$

Table 2.3.2-11. Total Cost for Upper Triangularization of an m x n dense matrix, m<n.

Operation	Cost
x & ÷	$(-1/3)m^3 + nm^2 + (n+16/3)m - 2n - 5$
+ & -	$(-1/3)m^3 + nm^2 + (14/6)m - n - 2$
√	2m - 2
sign	2m - 2

Table 2.3.2-12. Summary of Cost for Upper Triangularization of an m x n dense matrix, m>n.

Oper.	l=1	l=2	l=3	...	l=n-2	l=n-1	l=n	subtotal
x	m	m-1	m-2		m-n+3	m-n+2	m-n+1	(2m-n+1)n/2
+	m-1	m-2	m-3		m-n+2	m-n+1	m-n	(2m-n-1)n/2
√	1	1	1		1	1	1	n
x	1	1	1		1	1	1	n
+	1	1	1		1	1	1	n
√	1	1	1		1	1	1	n
÷	1	1	1		1	1	1	n
sign	1	1	1		1	1	1	n
x	m+1	m	m-1		m-n+4	m-n+3	m-n+2	(2m-n+3)n/2
+	1	1	1		1	1	1	n
sign	1	1	1		1	1	1	n
x	2m(n-1)	2(m-1)(n-2)	2(m-2)(n-3)		2(m-n+3)(2)	2(m-n+2)(1)	0	(1/3)n(n-1)(3m-n+2)
+	(m-1)(n-1)	(m-2)(n-2)	(m-3)(n-3)		(m-n+2)(2)	(m-n+1)(1)	0	Σ3
-	m(n-1)	(m-1)(n-2)	(m-2)(n-3)		(m-n+3)(2)	(m-n+2)(1)	0	Σ4

Σ3 + Σ4 = (2m-1)(n-1) +(2m-3)(n-2) + (2m-5)(n-3) + ... + (2m-2n+5)(2) + (2m-2n+3)(1) = n(n-1)[m-(1/3)n+1/6]

Table 2.3.2-13. Total Cost for Upper Triangularization of an m x n dense matrix, m>n.

Operation	Cost
x & ÷	n(2m-n+4) + (1/3)n(n-1)(3m-n+2)
+ & -	n(2m-n+3)/2 + n(n-1)[m-(1/3)n+1/6]
√	2n
sign	2n

As we observed, the cost of an upper triangularization by Householder reduction depends only on the dimension of the matrix; the structure of data within the matrix has no effect on the cost. As we will see in a later section, the structure of data greatly affects the cost of Fast Givens or Givens reduction.

The costs for triangularizing the Local Time Update, Local Measurement Update, Global Time Update, and Global Measurement Update system matrices are easily obtained by substituting the representative dimensional parameters into the cost equations of Tables 2.3.2-11 and 2.3.2-13. In substituting these parameters, it is important to note that the Local Time Update system matrix is of type $m<n$, while the other three system matrices are of type $m>n$.

2.3.2.1 Cost for a Local Time Update

This system matrix is m x n with $m<n$. Here, m represents $(q+n)$ and n represents $(q+n+1)$. Therefore, Table 2.3.2-11 is used to calculate the cost of triangularizing this system, and Table 2.3.2.1-1 shows the total cost.

Table 2.3.2.1-1. Total Cost for Upper Triangularization of an LTU.

Matrix dimensions: $[(q+n) \times (q+n+1)]$

Operation	Cost
x & +	$(1/3)[2(q+n)^3 + 6(q+n)^2 + 13(q+n) - 21]$
+ & -	$(1/3)[2(q+n)^3 + 3(q+n)^2 + 4(q+n) - 9]$
√	$2(q+n) - 1$
sign	$2(q+n) - 1$

2.3.2.2 Cost for a Local Measurement Update

This system matrix is m x n with $m>n$. Here, m represents $(n+m)$ and n represents $(n+1)$. Therefore, Table 2.3.2-13 is used to calculate the cost of triangularizing this system, and Table 2.3.2.1-1 shows the total cost.

Table 2.3.2.2-1. Total Cost for Upper Triangularization of an LMU.

Matrix dimensions: $[(n+m) \times (n+1)]$

Operation	Cost
x & ÷	$[n+1]\{2(n+m) - (n+1) + 4 + (1/3)n[3(n+m)-n+1]\}$
+ & -	$(n+1)(n+2m+2)/2 + n(n+1)[m+(1/6)(4n-1)]$
√	$2(n+1)$
sign	$2(n+1)$

2.3.2.3 Cost for a Global Time Update

This system matrix is m x n with m>n. Here, m represents (Mq+n) and n represents (q+n+1) in case A, while m represents [(M+1)q+n] in case B. Therefore, Table 2.3.2-13 is used to calculate the costs of triangularization in both cases A and B, and Tables 2.3.2.3-1 and 2.3.2.3-2 show the total costs for case A and case B respectively.

Table 2.3.2.3-1. Total Cost for Upper Triangularization of a GTU, Case A.

Matrix dimensions: $[(Mq+n) \times (q+n+1)]$

Operation	Cost
x & +	$(Mq+n)(q+n+1)(q+n+2) - (1/3)(q+n+1)[(a+n+1)^2 - 10]$
+ & -	$(1/3)(q+n+1)[3(Mq+n)(q+n+1) - (q+n+1)^2 + 4]$
$\surd$	$2(q+n+1)$
sign	$2(q+n+1)$

Table 2.3.2.3-2. Total Cost for Upper Triangularization of a GTU, Case B.

Matrix dimensions: $\{[(M+1)q+n] \times (q+n+1)\}$

Operation	Cost
x & ÷	$[(M+1)q+n]\,(q+n+1)(q+n+2) - (1/3)(q+n+1)\,[(q+n+1)^2 - 10]$
+ & -	$(1/3)(q+n+1)\{3[(M+1)q+n](q+n+1) - (q+n+1)^2 + 4\}$
$\surd$	$2(q+n+1)$
sign	$2(q+n+1)$

2.3.2.4 Cost for a Global Measurement Update

This system matrix is m x n with m>n. Here, m represents for (M+1)n and n represents (n+1). Therefore, Table 2.3.2-13 is used to calculate the costs of triangularization, and Table 2.3.2.4-1 shows the total cost.

Table 2.3.2.4-1. Total Cost for Upper Triangularization of a GMU.

Matrix dimensions: $[(M+1)n \times (n+1)]$

Operation	Cost
x & ÷	$(M+1)n(n+1)(n+2) - (1/3)(n+1)[(n+1)^2-10]$
+ & -	$(1/3)(n+1)[3(M+1)n(n+1)-(n+1)^2+4]$
$\surd$	$2(n+1)$
sign	$2(n+1)$

2.3.3 Description of the Givens Method

Householder transformations are very useful for introducing zeros on a grand scale, i.e., the annihilation of all but the first component of a vector. However, in many computations it is necessary to zero elements more selectively. Givens transformations are the tools for this application.

The QR factorization of a matrix, $A \in \mathbb{R}^{m\times n}$, can also be computed using the Givens method [3]. The Givens transformation matrix, also called the plane rotation matrix $G_{ij} \in \mathbb{R}^{m\times m}$, has the following form:

$$G_{ij} = \begin{bmatrix} 1 & & & & & & \\ & \cdot & & & & & \\ & & 1 & & & & \\ & & & \cos\phi_{ij} & \sin\phi_{ij} & & \\ & & & -\sin\phi_{ij} & \cos\phi_{ij} & & \\ & & & & & 1 & \\ & & & & & & \cdot \\ & & & & & & & \cdot \\ & & & & & & & & 1 \end{bmatrix}$$

with sine and cosine terms in the ith and jth rows and columns as shown above.

A given m x n matrix, A, can be upper triangularized in the following manner.

$$A_1 = G_{12}A = \begin{bmatrix} c_{12}a_1 + s_{12}a_2 \\ -s_{12}a_1 + c_{12}a_2 \\ a_3 \\ \cdot \\ \cdot \\ a_n \end{bmatrix}$$

where $c_{12} = \cos\phi_{12}$, $s_{12} = \sin\phi_{12}$, and a_i = ith row of A.

Choose ϕ_{12} such that $-s_{12}a_{11} + c_{12}a_{21} = 0$. This means that the first element in the second row of A_1 becomes zero. We don't actually calculate ϕ_{12} but only its sine and cosine values. The equation implies that $\tan\phi_{12} = a_{21}/a_{11}$. Therefore,

$$s_{12} = a_{21}/(a^2_{11} + a^2_{21})^{½},$$

$$c_{12} = a_{11}/(a^2_{11} + a^2_{21})^{½}$$

Thus, A_1 has a zero in the (2,1) position and the elements in the first two rows now differ, in general, from those of the given matrix A, while the remaining rows are the same.

To make the (3,1) position zero, we calculate $G_{13}A_1$, which produces A_2 and modifies the first and third rows of A_1 while leaving all others the same; the zero produced in the (2,1) position in the first stage remains zero. The angle ϕ_{13} is now chosen such that the (3,1) position of A_2 would be zero. Continuing in this fashion, zeroing the remaining elements in the first column, one after another, and then zeroing elements in the second column in the order (3,2), (4,2), ... , (m,2), and (4,3), (5,3), ... , (m,3) for the third column and so on. In all, assuming m > n, we will use (m-1) + (m-2) + ... + (m-n) plane rotation matrices and the result is that

$$GA = G_{n,m} \cdots G_{13}G_{12}A = R$$

is upper triangular. The matrices G_{ij} are all orthogonal so that G and G^{-1} are also orthogonal. With $Q = G^{-1}$ the given matrix A will be expressed as A = QR.

Based on the operation discussed above, we have a pseudocode for the Givens reduction as shown in Figure 2.3.3-1.

```
For k = 1 to min{m-1,n}
  For i = k+1 to m

    $s_{ki} = a_{ik}/(a^2_{kk} + a^2_{ik})^{1/2}$,

    $c_{ki} = a_{kk}/(a^2_{kk} + a^2_{ik})^{1/2}$

    $\hat{a}_k = c_{ki}a_k + s_{ki}a_i$

    $a_i = -s_{ki}a_k + c_{ki}a_i$

  End
End
```

Figure 2.3.3-1. Givens Reduction

Note that in the update of a_i it is the current a_k that is used, not the new a_k which has just been computed and is denoted by $\hat{a}_k$.

2.3.4 Description of the Fast Givens Method

As the name indicates, this method is derived from the Givens method. The calculations in the Givens reduction algorithm are rearranged so that they can be performed with "Householder speed" in principle. Following [5], the idea is to construct a matrix $M \in \mathbb{R}^{m \times m}$ such that

$$MA = S$$

is upper triangular and such that

$$MM^{tr} = D = \mathrm{diag}(d_1,\ldots,d_m), \quad d_i > 0$$

Since $D^{-1/2}M$ is orthogonal, it follows that

$$A = M^{-1}S = (D^{-1/2}M)^{-1}(D^{-1/2}S) = (M^{tr}D^{-1/2})(D^{-1/2}S)$$

is the QR factorization of A.

As we observed in the Givens reduction procedure, i.e., $G_{12}A = A_1$, only two rows of elements are altered in each transformation step. In this example, the first and second rows of A are altered. Therefore, the details of the computation can be explained at the 2-by-2 level. Let $x = (x_1,x_2)^{tr}$ and $D = \mathrm{diag}(d_1, d_2)$ where $d_1, d_2 > 0$, and define

$$M_1 = \begin{bmatrix} \beta_1 & 1 \\ 1 & \alpha_1 \end{bmatrix}$$

Observe that

$$M_1x = \begin{bmatrix} \beta_1 x_1 + x_2 \\ x_1 + \alpha_1 x_2 \end{bmatrix}$$

and

$$M_1DM_1^{tr} = \begin{bmatrix} d_2 + \beta_1 d^2_1 & d_1\beta_1 + d_2\alpha_1 \\ d_1\beta_1 + d_2\alpha_1 & d_1 + \alpha_1^2 d_2 \end{bmatrix}$$

If $x_2 \neq 0$ and $\alpha_1 = -x_1/x_2$ and $\beta_1 = -\alpha_1 d_2/d_1$, then

$$M_1 = \begin{bmatrix} x_2(1 + \gamma_1) \\ 0 \end{bmatrix}$$

$$M_1DM_1^{tr} = \begin{bmatrix} d_2(1 + \gamma_1) & 0 \\ 0 & d_1(1 + \gamma_1) \end{bmatrix},$$

where $\gamma_1 = -\alpha_1\beta_1 = (d_2/d_1)(x_1/x_2)^2$

Analogously, if we assume $x_1 \neq 0$ and define M_2 by

$$M_2 = \begin{bmatrix} 1 & \alpha_2 \\ \beta_2 & 1 \end{bmatrix} \qquad \beta_2 = -x_2/x_1, \quad \alpha_2 = -(d_1/d_2)\beta_2$$

then

$$M_2x = \begin{bmatrix} x_1(1 + \gamma_2) \\ 0 \end{bmatrix}$$

and

$$M_2DM_2^{tr} = \begin{bmatrix} d_1(1 + \gamma_2) & 0 \\ 0 & d_2(1 + \gamma_2) \end{bmatrix},$$

where $\gamma_2 = -\alpha_2\beta_2 = (d_1/d_2)(x_2/x_1)^2$.

Notice that the γ_i satisfy $\gamma_1\gamma_2 = 1$. Thus we can always select M_i in the above so that the "growth factor" $(1 + \gamma_i)$ is bounded by 2. Matrices of the form

$$M_1 = \begin{bmatrix} \beta_1 & 1 \\ 1 & \alpha_1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} 1 & \alpha_2 \\ \beta_2 & 1 \end{bmatrix}$$

satisfying $-1 \leq \alpha_i\beta_i \leq 0$ are referred to as Fast Givens transformations. Recalling the 2-by-2 Givens transformation matrix,

$$G_1 = \begin{bmatrix} \cos\phi_1 & \sin\phi_1 \\ -\sin\phi_1 & \cos\phi_1 \end{bmatrix},$$

we notice that premultiplication by a Fast Givens transformation involves about half the number of multiplies as premultiplication by a Givens transformation. As an example, let A be a 2 x 3 matrix. Then premultiplication by a Fast Givens transformation M_1A or M_2A, requires 6 multiplication and 6 addition operations, while 12 multiplication and 6 addition operations are needed by a Givens transformation. Another important observation is that the Fast Givens reduction does not require any square root calculations as the Givens reduction does. Therefore, the Fast Givens method is preferable to the "ordinary" Givens method, and only the Fast Givens method was investigated further.

The Fast Givens algorithm from [5] is shown below in Figure 2.3.4-1. It overwrites the matrix A with MA where MA is upper triangular and $MM^T = \mathrm{diag}(d_1,..,d_m)$.

```
$d_i := 1$  for $i = 1,...,m$
for $q = 2,...,m$
   for $p = 1,2,...,\min(q-1,n)$
      if $a_{qp} = 0$  then $\alpha = 0, \beta = 0$
         next p
      if $a_{qp} \neq 0$
         then
            $\alpha := -a_{pp}/a_{qp}$,  $\beta := -\alpha d_q/d_p$, $\gamma := -\alpha\beta$
            if $\gamma \leq 1$
               then
               $\begin{bmatrix} a_{pp} \cdots a_{pn} \\ a_{qp} \cdots a_{qn} \end{bmatrix} := \begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix} \begin{bmatrix} a_{pp} \cdots a_{pn} \\ a_{qp} \cdots a_{qn} \end{bmatrix}$
                  interchange $d_p$ and $d_q$.
                  $d_p := (1+\gamma)d_p$,  $d_q := (1+\gamma)d_q$
               else
                  interchange $\alpha$ and $\beta$.
                  $\alpha := 1/\alpha$,  $\beta := 1/\beta$,  $\gamma := 1/\gamma$

               $\begin{bmatrix} a_{pp} \cdots a_{pn} \\ a_{qp} \cdots a_{qn} \end{bmatrix} := \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} \begin{bmatrix} a_{pp} \cdots a_{pn} \\ a_{qp} \cdots a_{qn} \end{bmatrix}$

                  $d_p := (1+\gamma)d_p$,  $d_q := (1+\gamma)d_q$
   end
end
```

Figure 2.3.4-1. Fast Givens Algorithm

The pattern of annihilation by the algorithm is row order as shown below for a 6 x 5 matrix as an example (Figure 2.3.4-2).

x	x	x	x	x	1st row
1	x	x	x	x	2nd row
2	3	x	x	x	3rd row
4	5	6	x	x	4th row
7	8	9	10	x	5th row
11	12	12	14	15	6th row

Figure 2.3.4-2 Annihilation Pattern

The cost for upper triangularization of a matrix by Fast Givens reduction is obtained by counting the arithmetic operations in the algorithm shown in Figure 2.3.4-1. Note that instructions under the "else" statement in the algorithm require more arithmetic operations than those under the "then" statement by 3 counts of the division operation. We assume an equal probability of execution of those statements (else and then). We also assume that the overhead cost of communication between registers, initialization of memory and transfer is negligible compared with the arithmetic operation cost.

2.3.5 Computational Cost for Fast Givens Reduction

The cost calculations for the Fast Givens Algorithm of Figure 2.3.4-1 are shown in Tables 2.3.5-1 to 2.3.5-3. The elements marked as 1, 2, 3 in Figure 2.3.4-2 are annihilated in these tables.

Table 2.3.5-1. Annihilation of element marked as 1

$d_i := 1$ for $i = 1,\ldots,m$	
$q = 2$	; Annihilation of 2nd row and
$P = 1$	; 1st column position (marked as 1)
$\alpha = -a_{11}/a_{21}$	
$\beta = -\alpha \cdot d_2/d_1$	
$\tau = -\alpha \cdot \beta$	2 x, 2 ÷, 3 sign
if $\tau \leq 1$	1 -
$\begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \end{bmatrix}$	2n x, 2n +
$dp \leftrightarrow dq$	; interchange
$d_1 = (1+\tau)d_1$	
$d_2 = (1+\tau)d_2$	2 x, 2 +
else	
$\alpha \leftrightarrow \beta$	; interchange
$\alpha = 1/\alpha,\ \beta = 1/\beta,\ \tau = 1/\tau$	3 ÷
$\begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \end{bmatrix}$	2n x, 2n +
$d_1 = (1+\tau)d_1$	
$d_2 = (1+\tau)d_2$	2 x, 2 +

Table 2.3.5-2. Annihilation of element marked as 2

$q = 3$	; 3rd row
$P = 1$	; 1st column
$\alpha = -a_{11}/a_{31}$	
$ß = -\alpha \cdot d_3/d_1$	
$\tau = -\alpha \cdot ß$	2 x, 2 ÷, 3 sign
If $\tau \leq 1$	1 -
$\begin{bmatrix} ß & 1 \\ 1 & \alpha \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{31} & a_{32} & \cdots & a_{3n} \end{bmatrix}$	2n x, 2n +
$d_p \leftrightarrow d_q$	
$d_1 = (1+\tau)d_1$	
$d_3 = (1+\tau)d_3$	2 x, 2 +
else	
$\alpha \leftrightarrow ß$	
$\alpha = 1/\alpha$	
$ß = 1/ß$	
$\tau = 1/\tau$	3 ÷
$\begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{31} & a_{32} & \cdots & a_{3n} \end{bmatrix}$	2n x, 2n +
$d_1 = (1 + \tau)d_1$	
$d_3 = (1 + \tau)d_3$	2 x, 2 +

Table 2.3.5-3. Annihilation of element marked as 3

$q = 3$	; 3rd row
$p = 2$	; 2nd column
$\alpha = -a_{22}/a_{32}$	
$\beta = -\alpha \cdot d_3/d_2$	
$\tau = -\alpha \cdot \beta$	2 x, 2 ÷, 3 sign
If $\tau \leq 1$	1 -
$\begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix} \begin{bmatrix} a_{22} & a_{23} & \cdots & a_{2n} \\ a_{32} & a_{33} & \cdots & a_{3n} \end{bmatrix}$	2(n-1) x, 2(n-1) +
$d_p \leftrightarrow d_q$	
$d_2 = (1 + \tau)d_2$	
$d_3 = (1 + \tau)d_3$	2 x, 2 +
else	
$\alpha \leftrightarrow \beta$	
$\alpha = 1/\alpha$	
$\beta = 1/\beta$	
$\tau = 1/\tau$	3 ÷
$\begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} \begin{bmatrix} a_{22} & a_{23} & \cdots & a_{2n} \\ a_{32} & a_{33} & \cdots & a_{3n} \end{bmatrix}$	2 (n-1) x, 2(n-1) +
$d_2 = (1 + \tau)d_2$	
$d_3 = (1 + \tau)d_3$	2 x, 2 +

Now, we calculate the cost of upper triangularization of a dense m x n matrix. Recall that Fast Givens reduction is done by annihilating one element at a time as shown in Figure 2.3.4-2. Following the algorithm of (Figure 2.3.4-1), and collecting cost terms along arithmetic operations as in the above sample procedure, we have a summary of cost for matrix upper triangularization. The column data in the table represents the total cost of annihilation of all the elements in the respective row of the matrix. Tables 2.3.5-4 and 2.3.5-6 show the summary of cost for the cases $m<n$ and $m>n$ respectively. The total cost for triangularization is included in Tables 2.3.5-5 and 2.3.5-7 for $m<n$ and $m>n$ respectively. Here, we assume that the unit cost for a division and multiplication operation is the same, and the unit cost for subtraction and addition operations is the same.

Table 2.3.5-4. Summary of Cost for Upper Triangularization of an m x n dense matrix, m<n

Oper	q=2	q=3	q=4	....	q=m-1	q=m	Subtotal
x	2	2(2)	3(2)		(m-2)(2)	(m-1)(2)	Σ1
÷	2	2(2)	3(2)		(m-2)(2)	(m-1)(2)	Σ2
sign	3	2(3)	3(3)		(m-2)(3)	(m-1)(3)	Σ3
-	1	2(1)	3(1)		(m-2)(1)	(m-1)(1)	Σ4
x	2n	2n+2(n-1)	2n+2(n-1)+2(n-2)		2n+2(n-1)+ ··· +2(n-m+3)	2n+2(n-1)+ ··· +2(n-m+2)	Σ5
+	2n	2n+2(n-1)	2n+2(n-1)+2(n-2)		2n+2(n-1)+ ··· +2(n-m+3)	2n+2(n-1)+ ··· +2(n-m+2)	Σ6
x	2	2(2)	3(2)		(m-2)(2)	(m-1)(2)	Σ7
+	2	2(2)	3(2)		(m-2)(2)	(m-1)(2)	Σ8
÷	3/2	2(3/2)	3(3/2)		(m-2)(3/2)	(m-1)(3/2)	Σ9

$\Sigma1 = \Sigma2 = \Sigma7 = \Sigma8 = 2[1+2+3+\ldots+(m-1)] = m(m-1)$

$\Sigma3 = (3/2)m(m-1)$

$\Sigma4 = (1/2)m(m-1)$

$\Sigma5 = \Sigma6 = (m-1)(2n)+(m-2)[2(n-1)]+(m-3)[2(n-2)]+(m-4)[2(n-3)]+\ldots+(2)[2(n-m+3)]+(1)[2(n-m+2)]$

$= (-1/3)m^3 + (n+1)m^2 - (3n+2)m/3$

$\Sigma9 = (3/4)m(m-1)$

Table 2.3.5-5. Total Cost for Upper Triangularization of an m x n dense matrix, m<n

Operation	Cost
x & ÷	$(-1/3)m^3 + (n+19/4)m^2 - (12n+53)m/12$
+ & -	$(-1/3)m^3 + (n+5/2)m^2 - (6n+13)m/6$
sign	$(3/2)m(m-1)$

Table 2.3.5-6. Summary of Cost for Upper Triangularization of an m x n dense matrix, m>n

Oper	q=2	q=3	...	q=n	q=n+1	...	q=m	Subtotal
x	2	2(2)		(n-1)(2)	n(2)		n(2)	Σ1
÷	2	2(2)		(n-1)(2)	n(2)		n(2)	Σ2
sign	3	2(3)		(n-1)(3)	n(3)		n(3)	Σ3
-	1	2(1)		(n-1)(1)	n(1)		n(1)	Σ4
x	2n	2n+2(n-1)		2n+2(n-1)+ ··· +2(2)	2n+2(n-1)+ ··· +2(1)		2n+2(n-1)+..+2(1)	Σ5
+	2n	2n+2(n-1)		2n+2(n-1)+ ··· +2(2)	2n+2(n-1)+..+2(1)		2n+2(n-1)+..+2(1)	Σ6
x	2	2(2)		(n-1)(2)	n(2)		n(2)	Σ7
+	2	2(2)		(n-1)(2)	n(2)		n(2)	Σ8
÷	3/2	2(3/2)		(n-1)(3/2)	n(3/2)		n(3/2)	Σ9

$\Sigma 1 = \Sigma 2 = \Sigma 7 = \Sigma 8 = 2[1+2+3+\ldots+(m-1)]+(m-n)(2n) = n(2m-n-1)$

$\Sigma 3 = (3/2)n(2m-n-1)$

$\Sigma 4 = (1/2)n(2m-n-1)$

$\Sigma 5 = \Sigma 6 = [2n]+[2n+2(n-1)]+[2n+2(n-1)+2(n-2)]+\ldots+[2n+2(n-1)+\ldots+2(2)]+[2n+2(n-1)+\ldots+2(1)]$

$= n(n+1)(3m-n-2)/3$

$\Sigma 9 = (3/4)m(m-1)$

Table 2.3.5-7. Total Cost for Upper Triangularization of an m x n dense matrix, m>n

Operation	Cost
x & ÷	(15/4)n(2m-n-1) + n(n+1)(3m-n-2)/3
+ & -	(3/2)n(2m-n-1) + n(n+1)(3m-n-2)/3
sign	(3/2)n(2m-n-1)

2.3.5.1 Cost for a Local Time Update

The LTU matrix data structure is shown in Figure 2.3.5.1-1.

Matrix dimensions: $[(q+n) \times (q+n+1)]$

$$\begin{array}{c|ccc|cccccc|c}
 & & q & & & & n & & & & 1 \\
\hline
 & x & x & x & o & o & o & o & o & o & x \\
q & o & x & x & o & o & o & o & o & o & x \\
 & o & o & x & o & o & o & o & o & o & x \\
\hline
 & x & x & x & x & x & x & x & x & x & x \\
 & o & x & x & o & x & x & x & x & x & x \\
n & o & o & x & o & o & x & x & x & x & x \\
 & o & o & o & o & o & o & x & x & x & x \\
 & o & o & o & o & o & o & o & x & x & x \\
 & o & o & o & o & o & o & o & o & x & x
\end{array}$$

Figure 2.3.5.1-1. LTU Matrix Structure

The "x" marks in the figure represent non-zero data, and the "o" represents a zero value. As we see, the matrix is not dense. To upper triangularize this matrix, we select the non-zero elements ("x" marked) below the main diagonal and annihilate them one by one at a time. Table 2.3.5.1-1 shows a summary of cost required to annihilate all the elements below the main diagonal arranged along the row order.

Combining the cost terms for the multiplication and division together, and the cost terms for the addition and subtraction together and simplifying, we have a total cost for a LTU matrix upper triangularization in Table 2.3.5.1-2.

Table 2.3.5.1-1. Summary of Cost for Upper Triangularization of an LTU

Oper.	row q + 1	row q + 2	.....	row 2q	subtotal
x	q(2)	(q-1)(2)		(1)(2)	(q + 1)q
÷	q(2)	(q-1)(2)		(1)(2)	(q + 1)q
sign	q(3)	(q-1)(3)		(1)(3)	(3/2)(q + 1)q
-	q(1)	(q-1)(1)		(1)(1)	(1/2)(q + 1)q
x	2[(q + n + 1) + (q + n) + .. + (n + 2)]	2[(q + n + 1) + (q + n) + .. + (n + 2)]		2(n + 2)	$(1/3)q^3 + (n+2)q^2 + (n+5/3)q$
+	2[(q + n + 1) + (q + n) + .. + (n + 2)]	2[(q + n + 1) + (q + n) + .. + (n + 2)]		2(n + 2)	$(1/3)q^3 + (n+2)q^2 + (n+5/3)q$
x	q(2)	(q-1)(2)		(1)(2)	(q + 1)q
+	q(2)	(q-1)(2)		(1)(2)	(q + 1)q
÷	q(3/2)	(q-1)(3/2)		(1)(3/2)	(3/4)(q + 1)q

Table 2.3.5.1-2. Total Cost for Upper Triangularization of an LTU

Operation	Cost
x & ÷	$(1/3)q^3 + (n+23/4)q^2 + (n+65/12)q$
+ & -	$(1/3)q^3 + (n+7/2)q^2 + (n+19/6)q$
sign	(3/2)(q + 1)q

2.3.5.2 Cost for a Local Measurement Update

The LMU matrix data structure is shown in Figure 2.3.5.2-1.

Matrix dimensions: $[(n+m) \times (n+1)]$

$$\begin{array}{cccccccc}
 & & & n & & & & 1 \\
 & x & x & x & x & x & x & x \\
 & o & x & x & x & x & x & x \\
n & o & o & x & x & x & x & x \\
 & o & o & o & x & x & x & x \\
 & o & o & o & o & x & x & x \\
 & o & o & o & o & o & x & x \\
 & & & & & & & \\
 & x & x & x & x & x & x & x \\
m & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x
\end{array}$$

Figure 2.3.5.2-1. LMU Matrix Structure

The "x" marks in the figure represent non-zero data and the "o" represents a zero value. As we see, the matrix is not dense. To upper triangularize this matrix we select the non zero elements ("x" marked) below the main diagonal and annihilate them one by one at a time. Table 2.3.5.2-1 shows a summary of cost required to annihilate all the elements below the main diagonal arranged along the row order.

Combining the cost terms for the multiplication and division together, and the cost terms for addition and subtraction together and simplifying, we have a total cost for a LMU matrix upper triangularization in Table 2.3.5.2-2.

Table 2.3.5.2-1. Summary of Cost for Upper Triangularization of an LMU

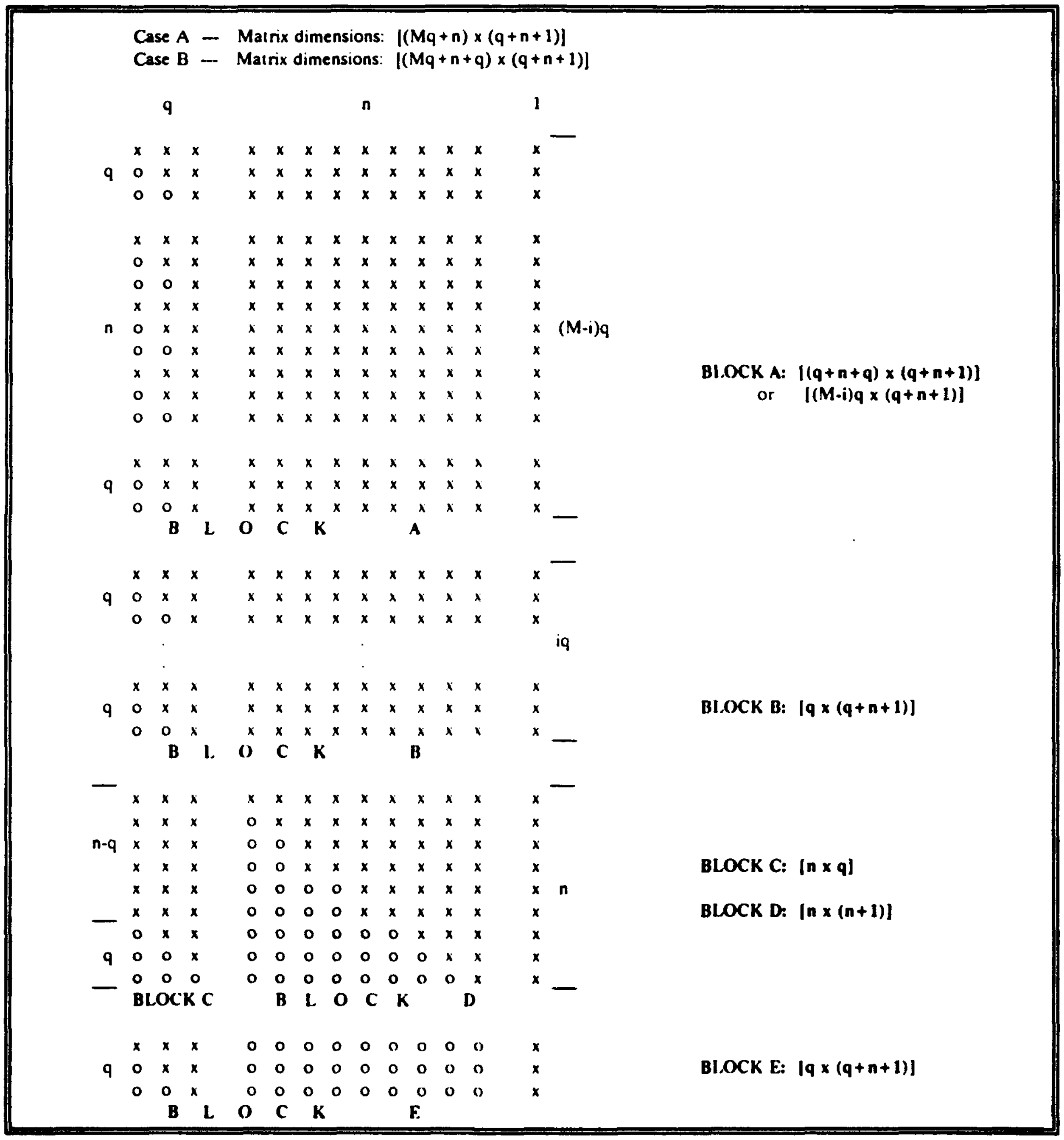
Oper.	row n+1	row n+2	.....	row n+m	subtotal
x	n(2)	(n+1)(2)		(n+1)(2)	2n+2(n+1)(m-1)
÷	n(2)	(n+1)(2)		(n+1)(2)	2n+2(n+1)(m-1)
sign	n(3)	(n+1)(3)		(n+1)(3)	3n+3(n+1)(m-1)
-	n(1)	(n+1)(1)		(n+1)(1)	n+(n+1)(m-1)
x	2(n+1)+2n+..+2(2)	2(n+1)+2n+..+2(2)+2(1)		2(n+1)+2n+...+2(2)+2(1)	(n+1)(n+2)m-2
+	2(n+1)+2n+..+2(2)	2(n+1)+2n+. +2(2)+2(1)		2(n+1)+2n+...+2(2)+2(1)	(n+1)(n+2)m-2
x	n(2)	(n+1)(2)		(n+1)(2)	2n+2(n+1)(m-1)
+	n(2)	(n+1)(2)		(n+1)(2)	2n+2(n+1)(m-1)
÷	n(3/2)	(n+1)(3/2)		(n+1)(3/2)	(3/2)n+(3/2)(n+1)(m-1)

Table 2.3.5.2-2. Total Cost for Upper Triangularization of an LMU

Operation	Cost
x & ÷	(15/2)[n+(n+1)(m-1)] + (n+1)(n+2)m - 2
+ & -	3[n+(n+1)(m-1)] + (n+1)(n+2)m - 2
sign	3[n+(n+1)(m-1)]

2.3.5.3 Cost for a Global Time Update

The GTU matrix data structure is shown in Figure 2.3.5.3-1. There are two cases of system formats (case A and case B). The difference between the two is that case B includes the extra data block labeled BLOCK E in Figure 2.3.5.3-1.



Case A — Matrix dimensions: [(Mq+n) x (q+n+1)]
Case B — Matrix dimensions: [(Mq+n+q) x (q+n+1)]
q
n
1
(M-i)q
BLOCK A
BLOCK A: [(q+n+q) x (q+n+1)] or [(M-i)q x (q+n+1)]
iq
BLOCK B
BLOCK B: [q x (q+n+1)]
n-q
BLOCK C
BLOCK D
BLOCK C: [n x q]
BLOCK D: [n x (n+1)]
BLOCK E
BLOCK E: [q x (q+n+1)]

Figure 2.3.5.3-1. GTU Matrix Structure

The "x" marks in the figure represent non-zero data and the "o" a zero value. As we see, the matrix is not dense. To upper triangularize this matrix we select the non-zero elements ("x" marked) below the main diagonal and annihilate them one by one at a time. Here, we make the assumptions that Mq is larger than q+n+1, i = M-5 for n = 3q, and i = M-4 for n = 2q. We first calculate the cost for an annihilation of each block and then combine all of the block costs according to each case. Table 2.3.5.3-1 shows the total cost for a triangularization of block A for both cases of n=3q and n=2q. The total annihilation cost for each block, B, C, D and E is included in Table 2.3.5.3-2.

So far we have obtained all the cost functions for each building block. Combining the cost functions from each block and sorting the resulting functions according to the variable q, we have the total costs for the system cases A and B in Tables 2.3.5.3-3 and 2.3.5.3-4 respectively. Note that there are i identical blocks of Block B type in the system (Global Time Update), where i was taken to be M-5 for n=3q, and M-4 for n=2q previously.

Table 2.3.5.3-1. Cost for Upper Triangularization of Block A

Operation	Cost for $n=3q$	Cost for $n=2q$
$\times$ & $\div$	$(-50/3)q^3 + (19n+401/4)q^2 + (3n+251/12)q - 2n - 19/2$	$(-17/3)q^3 + (11n+229/4)q^2 + (3n+227/12)q - 2n - 19/2$
$+$ & $-$	$(-50/3)q^3 + (19n+115/2)q^2 + (3n+85/6)q - 2n - 5$	$(-17/3)q^3 + (11n+65/2)q^2 + (3n+73/6)q - 2n - 5$
sign	$(57/2)q^2 + (9/2)q - 3$	$(33/2)q^2 + (9/2)q - 3$

Table 2.3.5.3-2. Cost of Filling Zeros into Blocks B, C, D and E

Operation	Cost for Block B	Cost for Block C
$\times$ & $\div$	$(1/3)q^3 + (n+23/4)q^2 + [n^2+(23/2)n+179/12]q$	$(-2/3)q^3 - (23/4)q^2 + [2n^2+(19/2)n-61/12]q$
$+$ & $-$	$(1/3)q^3 + (n+7/2)q^2 + (n^2+7n+49/6)q$	$(-2/3)q^3 - (7/2)q^2 + (2n^2+5n-17/6)q$
sign	$(3/2)q^2 + (3n+9/2)q$	$(-3/2)q^2 + (3n-3/2)q$

Operation	Cost for Block D	Cost for Block E
$\times$ & $\div$	$(1/3)n^3 + (23/4)n^2 + (179/12)n$	$(1/3)q^3 + (n+23/4)q^2 + (n+179/12)q$
$+$ & $-$	$(1/3)n^3 + (7/2)n^2 + (49/6)n$	$(1/3)q^3 + (n+7/2)q^2 + (n+49/6)q$
sign	$(3/2)n^2 + (9/2)n$	$(3/2)q^2 + (9/2)q$

Table 2.3.5.3-3. Total Cost for Upper Triangularization of a GTU --- Case A

Operation	Cost for n=3q	Cost for n=2q
x & ÷	$[(1/3)M-19]q^3 + [(M+14)n+(23/4)M+263/4]q^2$ $+ \{(M-3)n^2+[(23/2)M-45]n+(179/12)M-705/12\}q$ $+ [(1/3)n^3+(23/4)n^2+(155/12)n-19/2]$	$[(1/3)M-23/3]q^3 + [(M+7)n+(23/4)M+57/2]q^2$ $+ \{(M-2)n^2+[(23/2)M-67/2]n+(179/12)M-275/6\}q$ $+ [(1/3)n^3+(23/4)n^2+(155/12)n-19/2]$
+ & -	$[(1/3)M-19]q^3 + [(M+14)n+(7/2)M+73/2]q^2$ $+ [(M-3)n^2+(7M-27)n+(49/6)M-177/6]q$ $+ [(1/3)n^3+(7/2)n^2+(37/6)n-5]$	$[(1/3)M-23/3]q^3 + [(M+7)n+(7/2)M+15]q^2$ $+ [(M-2)n^2+(7M-20)n+(49/6)M-70/3]q$ $+ [(1/3)n^3+(7/2)n^2+(37/6)n-5]$
sign	$[(3/2)M+39/2]q^2 + [(3M-12)n+(9/2)M-39/2]q$ $+ [(3/2)n^2+(9/2)n-3]$	$[(3/2)M+9/2]q^2 + [(3M-9)n+(9/2)M-15]q$ $+ [(3/2)n^2+(9/2)n-3]$

Table 2.3.5.3-4. Total Cost for Upper Triangularization of a GTU --- Case B

Operation	Cost for n=3q	Cost for n=2q
x & ÷	$[(1/3)M-56/3]q^3 + [(M+15)n+(23/4)M+143/2]q^2$ $+ \{(M-3)n^2+[(23/2)M-44]n+(179/12)M-263/6\}q$ $+ [(1/3)n^3+(23/4)n^2+(155/12)n-19/2]$	$[(1/3)M-22/3]q^3 + [(M+8)n+(23/4)M+137/4]q^2$ $+ \{(M-2)n^2+[(23/2)M-65/2]n+(179/12)M-371/12\}q$ $+ [(1/3)n^3+(23/4)n^2+(155/12)n-19/2]$
+ & -	$[(1/3)M-56/3]q^3 + [(M+15)n+(7/2)M+40]q^2$ $+ [(M-3)n^2+(7M-26)n+(49/6)M-64/3]q$ $+ [(1/3)n^3+(7/2)n^2+(37/6)n-5]$	$[(1/3)M-22/3]q^3 + [(M+8)n+(7/2)M+37/2]q^2$ $+ [(M-2)n^2+(7M-19)n+(49/6)M-91/6]q$ $+ [(1/3)n^3+(7/2)n^2+(37/6)n-5]$
sign	$[(3/2)M+21]q^2 + [(3M-12)n+(9/2)M-15]q$ $+ [(3/2)n^2+(9/2)n-3]$	$[(3/2)M+21/2]q^2 + [(3M-9)n+(9/2)M-21/2]q$ $+ [(3/2)n^2+(9/2)n-3]$

2.3.5.4 Cost for a Global Measurement Update

The GMU matrix data structure is shown in Figure 2.3.5.4-1.

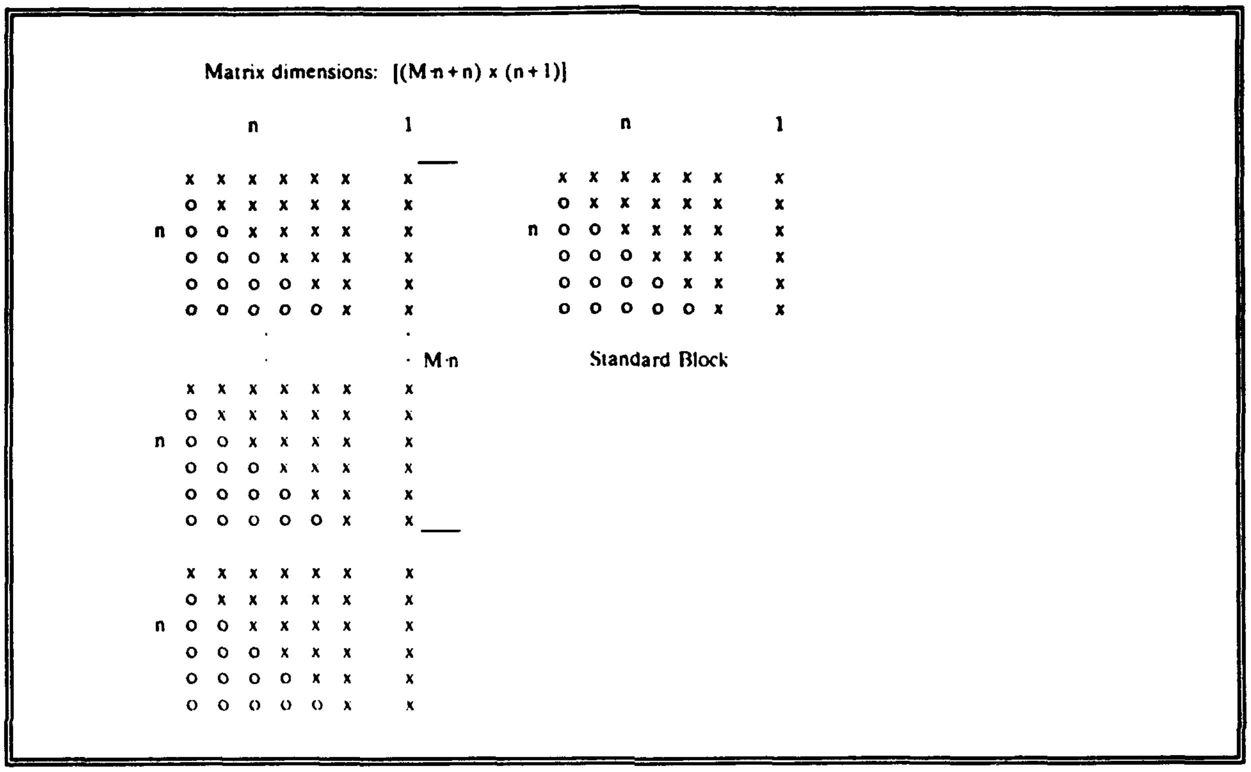


Figure 2.3.5.4-1. GMU Matrix Structure

The "x" marks in the figure represent non-zero data and the "o" a zero value. The total cost for upper triangularization of this GMU matrix is essentially the same as M times the cost for putting zeros into all of the non-zero elements ("x" marked) of the standard block. Again, applying the reduction procedure shown in Tables 2.3.5-1 to 2.3.5-3 for the cost calculation, we have a summary of cost for the standard block in Table 2.3.5.4-1. Table 2.3.5.4-2 shows the total cost for an upper triangularization of the GMU matrix. The element at the location of row n+1 and column n+1 will not be annihilated. Therefore, we subtract the cost for annihilation (filling a zero) of the element from the cost for M standard blocks.

Table 2.3.5.4-1. Cost for filling zeros into the standard block

Oper.	Row 1	Row 2	.....	Row n	Subtotal
x	(n+1)(2)	n(2)		2(2)	n(n+3)
÷	(n+1)(2)	n(2)		2(2)	n(n+3)
sign	(n+1)(3)	n(3)		2(3)	(3/2)n(n+3)
-	(n+1)(1)	n(1)		2(1)	(1/2)n(n+3)
x	2[(n+1)+n+...+1]	2[n+(n-1)+...+1]		2[2+1]	(1/3)[(n+1)(n+2)(n+3)-6]
+	2[(n+1)+n+...+1]	2[n+(n-1)+...+1]		2[2+1]	(1/3)[(n+1)(n+2)(n+3)-6]
x	(n+1)(2)	n(2)		2(2)	n(n+3)
+	(n+1)(2)	n(2)		2(2)	n(n+3)
÷	(n+1)(3/2)	n(3/2)		2(3/2)	(3/4)n(n+3)

Table 2.3.5.4-2. Total Cost for a Global Measurement Update

Operation	Cost
x & ÷	$M[(1/3)n^3+(23/4)n^2+(179/12)n] - 19/2$
+ & -	$M[(1/3)n^3+(7/2)n^2+(49/6)n] - 5$
sign	$M(3/2)n(n+3) - 3$

2.3.6 Comparison of Costs for Fast Givens and Householder Methods

So far we have obtained total cost functions for the four types of system matrices as well as for a general m by n dense matrix using the Fast Givens method and the Householder method. Table 2.3.6-1 shows a total cost comparison between two methods for a 30 by 10 dense matrix. The comparison on the four types of system matrices, LTU, LMU, GTU, GMU with the specific dimensions shown in the table is included in Table 2.3.6-2.

From the above comparison tables, we see that the Householder method is superior to the Fast Givens method for a general dense matrix. The Householder method, however, is inferior to the Fast Givens for the system matrices except the Global Time Update system. This is due to the system matrix structure being non sparse. In other words, the cost of Householder reduction is a function of matrix dimensions only, while the cost of Fast Givens reduction is a function of matrix dimensions and the matrix data structure. Another disadvantage of the Householder reduction method is that it requires square root operations while Fast Givens does not. Therefore, we adopt the Fast Givens method over the Householder method for the application at hand.

Table 2.3.6-1. Cost comparison between Fast Givens and Householder for an m x n dense matrix (m=30, n=10)

Operation	Fast Givens		Householder	
x & ÷	(15/4)n(2m-n-1) + n(n+1)(3m-n-2)/3	= 4698	n(2m-n+4) + (1/3)n(n-1)(3m-n+2)	= 3000
+ & -	(3/2)n(2m-n-1) + n(n+1)(3m-n-2)/3	= 3595	(2m-n+3)n/2 + n(n-1)[m-(1/3)n+1/6]	= 2680
sign	(3/2)n(2m-n-1)	= 735	2n	= 20
√	0	= 0	2n	= 20

Table 2.3.6-2. Cost Comparison between Fast Givens and Householder for the System Matrices

Local Time Update: [(q+n) x (q+n+1)] matrix
q=3, n=9

Operation	Fast Givens	Householder
x & ÷	185	1485
+ & -	158	1309
sign	18	22
√	0	22

Local Measurement Update: [(n+m) x (n+1)] matrix
n=9, m=3

Operation	Fast Givens	Householder
x & ÷	546	1020
+ & -	415	880
sign	87	20
√	0	20

Global Time Update: [(Mq+n) x (q+n+1)] matrix, Case A
M=10, q=3, n=9

Operation	Fast Givens	Householder
x & ÷	7308	6409
+ & -	5760	5876
sign	1032	26
√	0	26

Global Measurement Update: [(M+1)n x (n+1)] matrix
M=10, n=9

Operation	Fast Givens	Householder
x & ÷	8421	10590
+ & -	5995	9580
sign	1617	20
√	0	20

2.4 Matrix Downdating and Updating

The process of downdating/updating the transformed (triangularized) matrices is considered. This method is very useful because it drastically reduces the computational cost in matrix factorization (or triangularization). Rather than repeating the whole computing process for a new matrix which is formed by either adding new rows or deleting existing rows from an existing matrix, we can easily obtain a new transformed matrix for a fraction of the total cost by simply updating the existing transformed matrix.

2.4.1 Cost of Downdating

Downdating refers to a back process which removes the contribution made by the eliminated rows (data) from the original transformed matrix in a least squares problem.

Let A be a square matrix of order p and positive definite. Then A has a Cholesky factorization of the form

$$A = R^{tr}R$$

where R is an upper triangular matrix. Now, A is modified in a simple manner to form a new matrix A' whose Cholesky factorization is needed. Then we express A' in the following form,

$$A' = A - xx^{tr}$$

where x is a vector with a length of p which is removed from A. This modification is called downdating.

If $X \in \mathbb{R}^{nxp}$ and $y \in \mathbb{R}^{n}$ have the form

$$X = \begin{bmatrix} X' \\ x^{tr} \end{bmatrix}, \qquad y = \begin{bmatrix} y' \\ \eta \end{bmatrix}$$

then the downdating algorithm computes the factor corresponding to X' and y'; i.e., a least squares problem with a row removed. The row vector x^{tr} and a scalar η are removed together from X and y respectively. The relationship between matrix A and the matrix X can be explained with a least square problem. We wish to determine a b vector of length p such that

$$\rho^2 = \| y - Xb \|^2 \qquad \text{Eq. 1a}$$

is minimized (here $\| \bullet \|$ is the Euclidean vector norm). This problem can be solved by forming the matrix

$$(X,y)^{tr}(X,y) = \begin{bmatrix} A & c \\ ctr & \delta \end{bmatrix}$$

and computing its Cholesky factor

$$\begin{bmatrix} R & z \\ 0 & \rho \end{bmatrix}.$$

Then $Rb = z$ and ρ^2 is the residual sum of squares in Eq. 1a.

With this basic understanding, we review a subroutine program called "SCHDD" from LINPACK [6]. This subroutine downdates the Cholesky factorization $A = R^{tr}R$ of a positive matrix A to produce the Cholesky factorization $R'^{tr}R'$ of A' which is $A\text{-}xx^{tr}$, where x is a vector. Specifically, given an upper triangular matrix R of order p, a row vector x of length n, a column vector z of length n, and a scalar η, this subroutine SCHDD determines an orthogonal matrix U and a scalar ζ such that

$$U\begin{bmatrix} R & z \\ 0 & \zeta \end{bmatrix} = \begin{bmatrix} R' & z' \\ x^{tr} & \eta \end{bmatrix}$$

where R' is upper triangular. A residual norm ρ associated with z is downdated according to the formula $\rho' = \sqrt{(\rho^2 - \zeta^2)}$, if this is possible. If R and z have been obtained from the factorization of a least squares problem, then R' and z' are the factors corresponding to the problem with the observation (x,η) removed.

Using the subroutine "SCHDD", we calculate the cost of downdating one observation (a row vector) from the existing observations. The cost at each step of the subroutine is as follows:

1. Solve for A from the system, $R^{tr}A = X$ A; Vector

$$R = (n \times n) = \begin{matrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ 0 & 0 & r_{33} & \cdots & r_{3n} \\ 0 & 0 & 0 & \cdots & r_{4n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & 0 & \cdots & r_{nn} \end{matrix} \qquad z = \begin{matrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ \cdot \\ \cdot \\ z_n \end{matrix}$$

$$A = (1 \times n) = [\,a_1 \;\; a_2 \;\; a_3 \;\cdots\; a_n]$$
$$x = (1 \times n) = [\,x_1 \;\; x_2 \;\; x_3 \;\cdots\; x_n\,]$$

By back substitution, the vector A is obtained.

```
$s(1) \Leftarrow a_1 = x_1/r_{11}$                                                    $1 \div$

$s(2) \Leftarrow a_2 = (x_2 - r_{12}a_1)/r_{22}$                                        $1 \div, 1 \times, 1 -$

$s(3) \Leftarrow a_3 = (x_3 - r_{13}a_1 - r_{23}a_2)/r_{33}$                            $1 \div, 2 \times, 2 -$

$s(4) \Leftarrow a_4 = (x_4 - r_{14}a_1 - r_{24}a_2 - r_{34}a_3)/r_{44}$                $1 \div, 3 \times, 3 -$
    .                                                                                 .
    .                                                                                 .
$s(n) \Leftarrow a_n = (x_n - r_{1n}a_1 - r_{2n}a_2 - r_{3n}a_3 - \dots - r_{n-1,n}a_{n-1})/r_{nn}$  $1 \div, (n-1) \times, (n-1) -$

                                              Subtotal Cost: $n \div, n(n-1)/2 \times, n(n-1)/2 -$
```

```
2. Norm of A (or S):                                                  $n \times, n-1 +, 1 \surd$

   $Norm = \surd(a_1^2 + A_2^2 + \dots + A_n^2)$                         $n \times, (n-1) +, 1 \surd$
     if $Norm \geq 1$, then quit.

3. $\alpha = \surd(1 - Norm^2)$                                          $1 \times, 1 -, 1 \surd$

4. Determine the plane rotations (transformations).

       For ii = 1 to n
         $i = n - ii + 1$                                               $n +, n -$
         $scale = \alpha + |s(i)|$                                      $n +, n$ abs
         $pa = \alpha/scale$                                            $n \div$
         $pb = s(i)/scale$                                              $n \div$
         $Norm = \surd(pa^2 + pb^2)$                                    $2n \times, n +, n \surd$
         $c(i) = pa/Norm$                                               $n \div$
         $s(i) = pb/Norm$                                               $n \div$
         $\alpha = scale \cdot Norm$                                    $n \times$
       END

5. Apply the transformations to R. (To get R', where R' is a downdated R.)

   For j = 1 to n
     $xx = o$
     for ii = 1 to j
       $i = j - ii + 1$                                                 $n(n+1)/2 -, 2(n+1)/2 +$
       $T = c(i) \cdot xx + s(i)R(i,j)$                                 $n(n+1) \times, n(n+1)/2 +$
       $R(i,j) = c(i)R(i,j) - s(i) \cdot xx$                            $n(n+1) \times, n(n+1)/2 -$
       $xx = T$
     END
   END
```

6. Apply the transformations to z . (To get z', where z' is a downdated z.)

```
For j = 1 to nz        (nz is number of vectors to be downdated, nz=1 for our purpose)
  zeta = y(j)
    For i = 1 to n
      z(i,j) = [z(i,j) - s(i)zeta)/c(i)            n -, n x, n ÷
      zeta = c(i) x zeta - s(i)z(i,j)              n -, 2n x
    END
END
```

So far, we have collected step by step the costs above. Combining the cost terms along the arithmetic operations, we have a cost for a downdating with one observation in Table 2.4.1-1.

Table 2.4.1-1 Cost of downdating with one observation removed

Operation	Cost
x & ÷	n(5n+29)/2 + 1
+ & -	n(5n+17)/2
√	n + 2
sign (or abs)	n

where n is the order of the upper triangular matrix R.

The Global Measurement Update matrix consists of (M+1) standard matrices. The dimension of the standard matrix is [n x (n+1)]. The cost of downdating one standard block is now obtained. Since the standard block represents n observations (rows), the downdating cost with this standard block would be n times the cost obtained above for downdating one observation. Table 2.4.1-2 shows the cost of downdating with one standard block removed.

Table 2.4.1-2 Total Cost of downdating with a standard block removed

Operation	Cost
x & ÷	$(5/2)n^3 + (29/2)n^2 + n$
+ & -	$(5/2)n^3 + (17/2)n2$
√	$n^2 + 2n$
sign	n^2

2.4.2 Cost of Updating

The cost of updating, with one observation (a row vector) added, is obtained. Assume an observation vector whose length is n+1. Then the cost of updating with this vector will be the same as the cost of total annihilation of this observation vector. This cost can be easily obtained using the procedure in section 2.3.5.

The cost of updating the GMU system with a standard block is the same as the cost required to fill zeros into the block, and this cost was already shown in Table 2.3.5.4-1. Simplifying the table, we have a total cost for an updating with a standard block added as in Table 2.4.2-1. Note that the standard block is [n x (n+1)] in dimension.

Table 2.4.2-1 Cost of Updating with a Standard Block

Operation	Cost
x & ÷	$(1/3)n^3 + (23/4)n^2 + (179/12)n$
+ & -	$(1/3)n^3 + (7/2)n^2 + (49/6)n$
sign	$(3/2)n(n+3)$

2.4.3 Cost of Downdating and Updating

The total cost of downdating and updating with a standard block is the sum of the two costs obtained in sections 2.4.1 and 2.4.2. Table 2.4.3-1 shows the overall cost of downdating and updating with a standard block (n observations modified).

Table 2.4.3-1 Cost of Downdating and Updating with a standard block

Operation	Count
x & ÷	$(17/6)n^3 + (81/4)n^2 + (191/12)n$
+ & -	$(17/6)n^3 + 12n^2 + (49/6)n$
sign	$(5/2)n^2 + (9/2)n$
$\surd$	$n^2 + 2n$

2.5 Computational Cost for Matrix Upper Triangularization with Parallel Processing

Consider a dense matrix $A \in \mathbb{R}^{r \times c}$. Let R(i,j,k) denote a plane rotation in plane (i,j) which annihilates the element a_{ik} by the Fast Givens method, where $i \neq j$, $1 \le i, j \le r$ and $1 \le k \le c$. R(i,j,k) combines row i and row j such that $a'_{i,k} = 0$ in the resultant matrix A1.

$$A1 = \begin{bmatrix} a'_{j,k} & a'_{j,k+1} & \cdots & a'_{j,c} \\ a'_{i,k} & a'_{i,k+1} & \cdots & a'_{i,c} \end{bmatrix} \Leftarrow \begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix} \begin{bmatrix} a_{j,k} & a_{j,k+1} & \cdots & a_{j,c} \\ a_{i,k} & a_{i,k+1} & \cdots & a_{i,c} \end{bmatrix}$$

$$\text{or} \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} \begin{bmatrix} a_{j,k} & a_{j,k+1} & \cdots & a_{j,c} \\ a_{i,k} & a_{i,k+1} & \cdots & a_{i,c} \end{bmatrix}$$

The cost of applying R(i,j,k) via the Fast Givens method was already described in section 2.3.5 although it was not explicitly expressed in terms of parameters i, j, and k. The cost for R(i,j,k) is expressed in Table 2.5-1. It is noted that the cost is independent of parameters i and j which stand for row numbers of vectors (row vectors) involved in the operation (plane rotation). The cost rather depends on the column location of the element annihilated by this operation and the column dimension, c, of the matrix (or vectors).

Table 2.5-1 Cost for R(i,j,k)

Operation	Cost
x & ÷	2(c - k) + 19/2
+ & -	2(c - k) + 5
sign	3

2.5.1 Parallel Scheme for Application of R(i,j,k)

There are a few schemes for applying R(i,j,k) in parallel for matrix factorization (triangularization). Two schemes from [7] were reviewed.

One scheme, known as "Sameh and Kuck's", is systematic and assumes that j = i-1. Therefore, R(i,j,k) = R(i,i-1,k). In other words, the scheme always picks rows i and i-1 as a pair to annihilate (by a plane rotation) the element at the location of the ith row and kth column. Based on this rule, we can construct an annihilation pattern assuming that there are enough processors available for simultaneous plane rotations. Figure 2.5.1-1 shows the parallel annihilation scheme by Sameh and Kuck on a dense r x c matrix, where r=10 and c=6. The "x" marks represent elements above the main diagonal. The integer numbers indicate the step at which zeros are created (step of simultaneous annihilations). For instance, at step 1 we only perform R(10,9,1) to annihilate the element at (10,1). At step 9 however, we have as many as five simultaneous independent plane rotations, namely R(2,1,1), R(4,3,2), R(6,5,3), R(8,7,4), and R(10,9,5).

x	x	x	x	x	x	x	x
9	x	x	x	x	x	x	x
8	10	x	x	x	x	x	x
7	9	11	x	x	x	x	x
6	8	10	12	x	x	x	x
5	7	9	11	13	x	x	x
4	6	8	10	12	14	x	x
3	5	7	9	11	13	15	x
2	4	6	8	10	12	14	16
1	3	5	7	9	11	13	15

Figure 2.5.1-1. Sameh and Kuck's Parallel Scheme

It is easily seen that at step t we perform rotations R(i,i-1,k) such that $r+2k = i+t+1$, $1 \leq i \leq r$, where r stands for the row dimension of the matrix A (which is 10 in this example). Clearly, the total number of steps is $r+c-2$ if $r>c$ and $2c-3$ otherwise, where c stands for the column dimension of A (which is 6 in this case).

Another scheme, known as "Greedy", is shown in Figure 2.5.1-2.

x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x
3	6	x	x	x	x	x	x
2	5	8	x	x	x	x	x
2	4	7	10	x	x	x	x
1	4	6	9	11	x	x	x
1	3	5	8	10	12	x	x
1	3	5	7	9	11	13	x
1	2	4	6	8	10	12	14
1	2	3	5	7	9	11	13

Figure 2.5.1-2. Greedy Parallel Scheme

This scheme performs at each step as many rotations as possible, annihilating the elements in each column from bottom to top and in each row from left to right. For instance, at step 1 starting from the first column and the bottom (last) row, we can pick a maximum of five pairs of row vectors from this matrix for five simultaneous independent annihilations. This yields five zeros at the locations (10,1), (9,1), (8,1), (7,1), and (6,1) as marked with an integer "1" in the figure. At step 2 we can pick a maximum of two pairs of row vectors from the available top five rows for the annihilation of two elements at (4,1) and (5,1) while picking a maximum of two pairs of row vectors from the five available bottom rows for the annihilation of elements at (9,2) and (10,2). There are still two elements not annihilated in the first column at (3,1) and (2,1). At step 3 we can annihilate the element at (3,1) since we can pick a pair of row vectors from the available top three vectors. At the same time step, we can also pick two pairs of row vectors from the five row vectors available between the 4th row and 8th row to annihilate the elements at (7,2) and (8,2). Again, at this same time step 3, we are also able to pick another pair of row vectors to annihilate the element at (10,1). Expanding this idea, we can draw a parallel annihilation pattern called "Greedy" in Figure 2.5.1-2.

The Greedy method seems to yield a more optimum result than Sameh and Kuck's scheme for matrix triangularization. For instance, this method takes 14 steps while Sameh and Kuck's method takes 16 steps for completion of triangularization with the above 10 by 8 matrix. This method, however is not as systematic as Sameh and Kuck's. As we see, the pairing of row vectors for a plane rotation requires more complicated control than Sameh and Kuck's.

We apply these schemes on our system matrices assuming a full synchronization of the processors at the end of each step. Here a step means a set of independent rotations processed simultaneously by the processors. Let $R(i_1,j_1,k_1)$, $R(i_2,j_2,k_2)$, ... , $R(i_p,j_p,k_p)$, $p \leq r/2$, be the rotations performed at the nth step. The cost (time) needed to achieve this step will be the cost (time) for $R(i,j,k_{min})$, where $k_{min} = \min\{k1,k2,...,k_p\}$.

2.5.2 Cost of a Local Time Update

The LTU matrix structure is redrawn in Figure 2.5.2-1. The mark "x" denotes a non-zero element while the mark "o" denotes a zero element.

$$\begin{array}{c|ccc|cccccc|c}
 & & q & & & & n & & & & 1 \\
 & x & x & x & o & o & o & o & o & o & x \\
q & o & x & x & o & o & o & o & o & o & x \\
 & o & o & x & o & o & o & o & o & o & x \\
 & x & x & x & x & x & x & x & x & x & x \\
 & o & x & x & o & x & x & x & x & x & x \\
n & o & o & x & o & o & x & x & x & x & x \\
 & o & o & o & o & o & o & x & x & x & x \\
 & o & o & o & o & o & o & o & x & x & x \\
 & o & o & o & o & o & o & o & o & x & x
\end{array}$$

Figure 2.5.2-1. LTU Matrix Structure

There are q(q+1)/2 non-zero elements under the main diagonal which are to be annihilated. Applying Sameh and Kuck's scheme, we have the following annihilation pattern drawn for the q = 3 case in Figure 2.5.2-2. The Greedy method also yields the same result for the q = 3 case. For a larger q, two methods are expected to produce slightly different results but neither method seems to yield a reasonable optimum solution for a parallel process. This is due to the unique data structure of the LTU.

$$\begin{array}{c|ccc|cccccc|c}
 & & q & & & & n & & & & 1 \\
 & x & x & x & o & o & o & o & o & o & x \\
q & o & x & x & o & o & o & o & o & o & x \\
 & o & o & x & o & o & o & o & o & o & x \\
 & 1 & 3 & 5 & x & x & x & x & x & x & x \\
 & o & 2 & 4 & o & x & x & x & x & x & x \\
n & o & o & 3 & o & o & x & x & x & x & x \\
 & o & o & o & o & o & o & x & x & x & x \\
 & o & o & o & o & o & o & o & x & x & x \\
 & o & o & o & o & o & o & o & o & x & x
\end{array}$$

Figure 2.5.2-2. Sameh and Kuck's Scheme for an LTU (q=3 case)

One possible optimum solution (pattern) is presented in Table 2.5.2-3. This is based on the realization that we can pair vectors in the following fashion, row 1 and row q+1, row 2 and row q+2, ... , and row q and row 2q, simultaneously at every time step.

$$\begin{array}{c|ccc|cccccc|c}
 & & q & & & & n & & & & 1 \\
 & x & x & x & o & o & o & o & o & o & x \\
q & o & x & x & o & o & o & o & o & o & x \\
 & o & o & x & o & o & o & o & o & o & x \\
 & 1 & 2 & 3 & x & x & x & x & x & x & x \\
 & o & 1 & 2 & o & x & x & x & x & x & x \\
n & o & o & 1 & o & o & x & x & x & x & x \\
 & o & o & o & o & o & o & x & x & x & x \\
 & o & o & o & o & o & o & o & x & x & x \\
 & o & o & o & o & o & o & o & o & x & x
\end{array}$$

Figure 2.5.2-3. Optimum Annihilation Pattern (q=3 case)

Based on this optimum pattern, the cost for an upper triangularization in this parallel process is evaluated. The step by step cost is listed as follows:

Step	Cost in terms of R(i,j,k)
1	R(q+1,1,1)
2	R(q+1,2,2)
.	.
.	.
q	R(q+1,q,q)

The cost for R(i,j,k) was already obtained in Table 2.5-1. Converting the cost in R(...) terms to an actual cost using Table 2.5-1, and adding the costs along arithmetic operations we have a total cost in Table 2.5.2-1.

Table 2.5.2-1. Total Cost for a Local Time Update in a Parallel Process

Oper.	Cost
x & ÷	$[2(c-1)+19/2] + [2(c-2)+19/2] + \cdots + [2(c-q)+(19/2)] = q[2c+(17/2)] - q^2$
+ & -	$[2(c-1)+5] + [2(c-2)+5] + \cdots + [2(c-q)+5] = q(2c+4) - q^2$
sign	$3q$
	Where $c = q+n+1$, q and n are dimension parameters. See Figure 2.5.2-1.

2.5.3 Cost of a Local Measurement Update

The LMU matrix structure is redrawn in Figure 2.5.3-1. The mark "x" denotes a non-zero element while the mark "o" denotes a zero element.

$$\begin{array}{c|cccccc|c}
 & & & n & & & & 1 \\
 & x & x & x & x & x & x & x \\
 & o & x & x & x & x & x & x \\
n & o & o & x & x & x & x & x \\
 & o & o & o & x & x & x & x \\
 & o & o & o & o & x & x & x \\
 & o & o & o & o & o & x & x \\
 & x & x & x & x & x & x & x \\
m & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x
\end{array}$$

Figure 2.5.3-1. LMU Matrix Structure

We apply the Greedy scheme first, but it is too complicated to derive a general cost function in terms of the parameters, m and n in this parallel scheme. Instead, we consider two typical system cases of m equal to 2 and 3.

For m = 3, we have a parallel annihilation pattern in Figure 2.5.3-2. The figure is drawn for the case n = 6 as an example.

$$\begin{array}{c|ccccccc}
 & & & n & & & & 1 \\
 & x & x & x & x & x & x & x \\
 & o & x & x & x & x & x & x \\
n & o & o & x & x & x & x & x \\
 & o & o & o & x & x & x & x \\
 & o & o & o & o & x & x & x \\
 & o & o & o & o & o & x & x \\
 & & & & & & & \\
 & 2 & 4 & 6 & 8 & 10 & 12 & x \\
m & 1 & 3 & 5 & 7 & 9 & 11 & 13 \\
 & 1 & 2 & 4 & 6 & 8 & 10 & 12
\end{array}$$

Figure 2.5.3-2. Greedy Scheme for an LMU (m=3, n=6 case)

The step-by-step cost for an upper triangularization of an LMU employing the Greedy scheme is as follows:

Step	Cost in terms of R (i,j,k)
1	R(n+m,n+m-2,1)
2	R(n+m-2,1,1)
3	R(n+m-1,n+m-2,2)
4	R(n+m-2,2,2)
.	.
.	.
2n-1	R(n+m-1,n+m-2,n)
2n	R(n+m-2,n,n)
2n+1	R(n+m-1,n+m-2,n+1)

Using Table 2.5-1 for the cost of R(i,j,k) and adding all the terms above, we have a total cost for an LMU (m=3) in Table 2.5.3-1. The parameter "c" in Table 2.5-1 is equivalent to n+1 in the LMU system.

Table 2.5.3-1 Total Cost of an LMU using Greedy Scheme (m=3)

Operation	Cost
x & ÷	$2n^2 + 21n + 19/2$
+ & -	$2n^2 + 12n + 5$
sign	$6n + 3$

For m = 2, we have a parallel annihilation pattern in Figure 2.5.3-3. The figure is drawn for n = 6 as an example.

$$
\begin{array}{c|cccccccc}
 & & & n & & & & & 1 \\
 & x & x & x & x & x & x & & x \\
 & o & x & x & x & x & x & & x \\
n & o & o & x & x & x & x & & x \\
 & o & o & o & x & x & x & & x \\
 & o & o & o & o & x & x & & x \\
 & o & o & o & o & o & x & & x \\
 & & & & & & & & \\
 & 2 & 4 & 6 & 8 & 10 & 12 & & x \\
m & 1 & 3 & 5 & 7 & 9 & 11 & & 13
\end{array}
$$

Figure 2.5.3-3. Greedy Scheme for an LMU (m=2, n=6 case)

As we see in figure 2.5.3-3, the Greedy scheme does not yield a parallel process. Therefore, the total cost for this LMU system with m=2 will be the same as the cost obtained for a sequential process in Table 2.3.5.2-2 of section 2.3.5.2. Note that the total cost for an LMU with m=3 using Greedy's parallel scheme is the same as this total cost.

Figure 2.5.3-4 shows the parallel annihilation pattern which results from using Sameh & Kuck's scheme on the LMU system. Only the area annihilated by this process is redrawn with an expansion. Here we must realize that Sameh and Kuck's rule, R(i,j,k) = R(i,i-1,k) cannot be applied for annihilation of elements from (n+1,1) to (n+1,n-1) at row n+1 because all elements of nth row below the main diagonal are zero. Therefore, the rule for pairing of vectors is modified for those elements; for example, R(n+1,1,1) rather than R(n+1,n,1) for annihilation of the element at (n+1,1).

$$
\begin{array}{c|cccccc}
 & & & n & & & 1 \\
 & m & m+2 & m+4 & m+6 & .. & x \\
 & m-1 & m+1 & m+3 & m+5 & .. & m-1+2n \\
m & . & . & . & . & .. & . \\
 & . & . & . & . & .. & . \\
 & 3 & 5 & 7 & 9 & .. & 3+2n \\
 & 2 & 4 & 6 & 8 & .. & 2+2n \\
 & 1 & 3 & 5 & 7 & .. & 1+2n
\end{array}
$$

Figure 2.5.3-4. Sameh and Kuck's scheme for an LMU, where an integer number indicates the annihilation step and the mark x stands for an element which may not be annihilated.

Again collecting the cost term step by step in the form of R(i,j,k), we have the following:

Step	Cost in terms of R(i,j,k)
1	R(n+m,n+m-1,1)
2	R(n+m-1,n+m-2,1)
3	R(n+m-2,n+m-3,1)
.	.
.	.
m	R(n+1,1,1)
m+1	R(n+2,n+1,2)
m+2	R(n+1,2,2)
m+3	R(n+2,n+1,3)
m+4	R(n+1,3,3)
.	.
.	.
m-1+2n	R(n+2,n+1,n+1)

Referring to Table 2.5-1 for a cost conversion, and adding the cost for all steps above, we have the total cost in Table 2.5.3-2. Note that the parameter c in Table 2.5-1 is equivalent to n+1 for the LMU system.

Table 2.5.3-2 Total Cost of an LMU using Sameh and Kuck's Scheme

Operation	Cost
x & ÷	2n(m+n+17/2) + (19/2)(m-1)
+ & -	2n(m+n+4) + 5(m-1)
sign	3(m-1+2n)

Where m and n are dimension parameters. Refer to Figure 2.5.3-1.

2.5.4 Cost of a Global Time Update

For the purpose of cost evaluation, the system matrix is rearranged and shown in Figure 2.5.4-1.

$$
\begin{array}{c|ccc|ccccccccc|c|l}
 & & q & & & & & & n & & & & & 1 & \\
 & x & x & x & x & x & x & x & x & x & x & x & x & x & \\
q & o & x & x & x & x & x & x & x & x & x & x & x & x & \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
\hline
 & x & x & x & x & x & x & x & x & x & x & x & x & x & \\
 & x & x & x & o & x & x & x & x & x & x & x & x & x & \\
n-q & x & x & x & o & o & x & x & x & x & x & x & x & x & \\
 & x & x & x & o & o & o & x & x & x & x & x & x & x & \\
 & x & x & x & o & o & o & o & x & x & x & x & x & x & n \\
 & x & x & x & o & o & o & o & o & x & x & x & x & x & \\
\hline
 & o & x & x & o & o & o & o & o & o & x & x & x & x & \\
q & o & o & x & o & o & o & o & o & o & o & x & x & x & \\
 & o & o & o & o & o & o & o & o & o & o & o & x & x & \\
\hline
 & x & x & x & x & x & x & x & x & x & x & x & x & x & \\
q & o & x & x & x & x & x & x & x & x & x & x & x & x & \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
 & & \vdots & & & & & & \vdots & & & & & \vdots & (M-1)q \\
 & x & x & x & x & x & x & x & x & x & x & x & x & x & \\
q & o & x & x & x & x & x & x & x & x & x & x & x & x & \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
\hline
 & x & x & x & o & o & o & o & o & o & o & o & o & x & \\
q & o & x & x & o & o & o & o & o & o & o & o & o & x & \text{Block required for case B} \\
 & o & o & x & o & o & o & o & o & o & o & o & o & x & \\
\end{array}
$$

Figure 2.5.4-1. GTU Matrix Structure (Rearranged)

Case A System:

The system matrix for case A is of dimension {[q+n+(M-1)q] x [q+n+1]}. Although the Greedy scheme will produce a more efficient result than Sameh and Kuck's in annihilation, it is too complicated to derive a general cost function for the Greedy scheme. Nonetheless, we attempt to calculate a cost for Sameh & Kuck's scheme as follows. Revisiting section 2.5.1 which describes Sameh and Kuck's parallel scheme, we draw a 10 by 8 rectangular matrix which shows the parallel scheme with key locations marked with a # sign. The key location gives the maximum cost of annihilation among the elements involved in a given annihilation step. This is due to the nature of the cost function for R(i,j,k) and the synchronized operation of processors. Again it is restated that the cost for an R(i,j,k) is a function of the k parameter only; neither i nor j affects the cost.

x	x	x	x	x	x	x	x
9#	x	x	x	x	x	x	x
8#	10#	x	x	x	x	x	x
7#	9	11#	x	x	x	x	x
6#	8	10	12#	x	x	x	x
5#	7	9	11	13#	x	x	x
4#	6	8	10	12	14#	x	x
3#	5	7	9	11	13	15#	x
2#	4	6	8	10	12	14	16#
1#	3	5	7	9	11	13	15

Figure 2.5.4-2. Sameh and Kuck's Parallel Scheme with Key Locations

Therefore, the overall cost for an upper triangularization of the matrix under this parallel scheme is the sum of the costs for annihilation of the elements marked with a # sign behind the step number in the figure.

$$\begin{aligned}\text{Total Cost} &= \text{Cost}[\, R(10,9,1) + R(9,8,1) + R(8,7,1) + R(7,6,1) + R(6,5,1) \\ &\quad + R(5,4,1) + R(4,3,1) + R(3,2,1) + R(2,1,1) + R(3,2,2) + R(4,3,3) \\ &\quad + R(5,4,4) + R(6,5,5) + R(7,6,6) + R(8,7,7) + R(9,8,8)\,] \\ &= \text{Cost}[\, 9 \cdot R(10,9,1) + R(3,2,2) + R(4,3,3) + R(5,4,4) + R(6,5,5) \\ &\quad + R(7,6,6) + R(8,7,7) + R(9,8,8)\,]\end{aligned}$$

Applying this idea to the system matrix (Case A), the key locations are marked as shown in Figure 2.5.4-3. The overall cost of annihilation then is obtained by counting the marked (#) locations along the column number. Although the marked key locations are obtained fairly easily for the rectangular matrix shown in Figure 2.5.4-2, it is not obvious how many locations should be marked on each q x (q+n+1) block for the general case of q and n. The right most or left most column location will be obtained by solving linear equations. From the above Figure 2.5.4-2, it is noted that all the locations annihilated in a given step lie on a straight line of slope -2. This is an important fact of Sameh and Kuck's scheme and will be applied in finding key locations for the system.

$$\begin{array}{c|ccc|ccccccccc|c|l}
 & & q & & & & & & n & & & & & 1 & \\
 & x & x & x & x & x & x & x & x & x & x & x & x & x & \\
q & o & x & x & x & x & x & x & x & x & x & x & x & x & 1^{st}\ [q \times (q+n+1)]\ \text{Block} \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
\hline
 & x\# & x\# & x\# & x & x & x & x & x & x & x & x & x & x & \\
 & x\# & x\# & x\# & o & x & x & x & x & x & x & x & x & x & \\
n-q & x\# & x & x & o & o & x & x & x & x & x & x & x & x & \\
 & x\# & x & x & o & o & o & x & x & x & x & x & x & x & \\
 & x\# & x & x & o & o & o & o & x & x & x & x & x & x\ \ n & [n \times (q+n+1)]\ \text{Block} \\
 & x\# & x & x & o & o & o & o & o & x & x & x & x & x & \\
\hline
 & o & x & x & o & o & o & o & o & o & x & x & x & x & \\
q & o & o & x & o & o & o & o & o & o & o & x & x & x & \\
 & o & o & o & o & o & o & o & o & o & o & o & x & x & \\
\hline
 & x\# & x\# & x & x & x & x & x & x\# & x\# & x\# & x\# & x\# & x & \\
q & o & x\# & x\# & x & x & x & x & x & x\# & x\# & x\# & x\# & x\# & 2^{nd}\ [q \times (q+n+1)]\ \text{Block} \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
\hline
 & x\# & x\# & x & x & x & x & x & x & x & x & x & x & x & \\
q & o & x\# & x & x & x & x & x & x & x & x & x & x & x & 3^{rd}\ [q \times (q+n+1)]\ \text{Block} \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
 & & \cdot & & & & & & \cdot & & & & & \cdot & \\
 & & \cdot & & & & & & \cdot & & & & & \cdot\ (M-2)q & \\
 & x\# & x\# & x & x & x & x & x & x & x & x & x & x & x & \\
q & o & x\# & x & x & x & x & x & x & x & x & x & x & x & M^{th}\ [q \times (q+n+1)]\ \text{Block} \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
\end{array}$$

Figure 2.5.4-3. GTU Matrix (Case A) with Marked Key Locations

Finding the key locations in the [n x (q+n+1)] block is straightforward. However, the key locations in the 2nd [q x (q+n+1)] block are not as obvious, especially for those on rows q+n+1 and q+n+2. We solve for those locations by solving linear equations based on the fact that the slope of the linear equation should be -2 as mentioned above. The area of the [n x (q+n+1)] block and 2nd [q x (q+n+1)] block is redrawn in Figure 2.5.4-4 to indicate critical locations on the top two rows of the 2nd [q x (q+n+1)] block.

$$\begin{array}{c|ccc|ccccccccc|c|l}
 & & q & & & & & & n & & & & & 1 & \\
\hline
 & x\# & x\# & x\# & x & x & x & x & x & x & x & x & x & x & \\
 & x\# & x\# & x\# & o & x & x & x & x & x & x & x & x & x & \\
n-q & x\# & x & x & o & o & x & x & x & x & x & x & x & x & \\
 & x\# & x & x & o & o & o & x & x & x & x & x & x & x & \\
 & x\# & x & x & o & o & o & o & x & x & x & x & x & x\ \ n & [n \times (q+n+1)]\ \text{Block} \\
 & x\# & x & x & o & o & o & o & o & x & x & x & x & x & \\
\hline
 & o & x & x & o & o & o & o & o & o & x & x & x & x & \\
q & o & o & x & o & o & o & o & o & o & o & x & x & x & \\
 & o & o & o & o & o & o & o & o & o & o & o & x & x & \\
\hline
 & x\# & R1 & x & x & x & x & x & L1 & x\# & x\# & x\# & x\# & x & \\
q & o & x\# & R2 & x & x & x & x & x & L2 & x\# & x\# & x\# & x\# & 2^{nd}\ [q \times (q+n+1)]\ \text{Block} \\
 & o & o & x & x & x & x & x & x & x & x & x & x & x & \\
\end{array}$$

Figure 2.5.4-4. Critical Locations in 2nd [q x (q+n+1)] Block

Critical locations are labeled with the letters **R1**, **R2**, **L1**, and **L2** in the figure.

We solve for the exact column locations of the spots labeled **L1** and **L2**. Imagine a coordinate system (c for a horizontal axis and r for a vertical axis) and superimpose the coordinate system onto the above figure, such that the far bottom left element in the figure coincides with the coordinates (1,1) in the coordinate system. Then we can set up the following linear equations for **L1** and **L2**.

For **L1**, we have
$$r = -2[c-(q+1)] + (n+q-1) \quad \text{Eq. 1}$$
$$r = q \quad \text{Eq. 2}$$
$$r = q - 1 \quad \text{Eq. 3}$$

Solving equations 1 and 2 simultaneously for the unknown c,
$$q = -2[c-(q+1)] + (n+q-1)$$

then c is found to be $c = (2q+n+1)/2.$

Since c must be an integer, we round up the resultant real value to its closest integer.

Therefore, $c = RU[(2q+n+1)/2],$

where RU stands for a round up of argument. This c value represents the column location of **L1**.

For **L2**, we solve equations 1 and 3 simultaneously. Then we have,

$$q-1 = 2[c-(q+1)] + (n+q-1)$$

and c is found to be $c = (2q+n+2)/2.$

Since c must be an integer, we round up the resultant real value to its closest integer.

Therefore, $c = RU[(2q+n+2)/2]$

This value of c represents the column location of **L2**.

Now we solve for the column locations of **R1** and **R2**.

For **R1**, $r = -2(c-1) + 2q \quad \text{Eq. 4}$

Solving equations 2 and 4 simultaneously for the unknown c, we have

$$q = -2(c-1) + 2q,$$

then c is found to be $\quad c = (q+2)/2.$

Since c must be an integer, we round down the resultant real value to its closest integer.

Therefore, $\quad c = RD[(q+2)/2],$

where RD stands for a round down of argument. This c value represents the column location of **R1**.

For **R2**, we solve equations 4 and 3 simultaneously. Then we have,

$$q-1 = -2(c-1) + 2q$$

and c is found to be $\quad c = (q+3)/2.$

Since c must be an integer, we round down the resulting real value to its closest integer.

Therefore, $\quad c = RD[(q+3)/2]$

This c value represents the column location of **R2**.

Now, we need to find the critical locations in the 3rd [q x (q+n+1)] block. The area of the 2nd and 3rd [q x (q+n+1)] blocks are redrawn in Figure 2.5.4-5 to indicate the critical locations in the 3rd block.

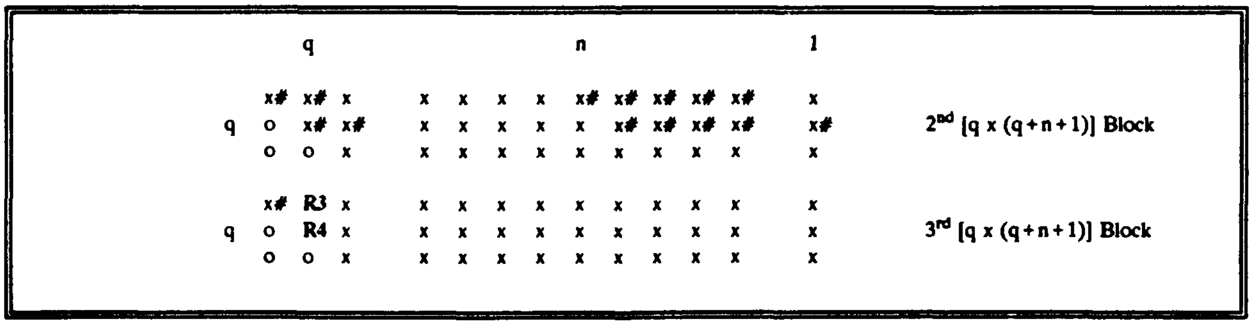


Figure 2.5.4-5. Critical Locations in 3rd [q x (q+n+1)] Block

The column locations of the critical spots labeled with **R3** and **R4** are obtained in a similar way. Again the same coordinate system is superimposed on Figure 2.5.4-4 such that the far bottom left element of the figure coincides with the coordinates (1,1) in the coordinate system. Then we can set up equations for the column locations of the critical spots **R3** and **R4**.

For **R3**, we have $r = -2(c-1) + (2q-1)$ Eq. 5

Solving these two equations simultaneously for the unknown c, we find $c = (q+1)/2$.

Since c must be an integer, we round down the resultant real value to its closest integer. Then we have,

$$c = RD[(q+1)/2],$$

where RD stands for a round down of argument. This value of c represents the column location of **R3**.

For **R4**, we solve equations 5 and 3 simultaneously. Then we have,

$$q-1 = -2(c-1) + (2q-1)$$

and c is found to be

$$c = (q+2)/2.$$

Since c must be an integer, we round down the resulting real value to its closest integer.

Therefore, $c = RD[(q+2)/2]$

This value of c represents the column location of **R4**. Note that the column locations we obtained for **R3** and **R4** are also applicable to the 4th through Mth $[q \times (n+q+1)]$ block.

The total cost for the GTU system (Case A), is the sum of each block cost as follows:

1) Cost of $[n \times (q+n+1)]$ Block

Counting the key marked locations and adding the individual cost required to annihilate the key located elements, we have the following cost equation,

$$\text{Cost} = (n-q)R(i,j,1) + 2R(i,j,2) + 2R(i,j,3) + \ldots + 2R(i,j,q)$$

where i and j are arbitrary and do not affect the cost.

Converting this equation using Table 2.5-1, we present the total cost in Table 2.5.4-1. The parameter in Table 2.5-1 is equivalent to q+n+1 for this block.

Table 2.5.4-1. Cost of [n x (q+n+1)] Block

Operation	Count
x & ÷	$2n^2 + (4q+11/2)n + (15/2)q - 19$
+ & -	$2n^2 + (4q+1)n + 3q - 10$
sign	$3n + 3q - 6$

2) Cost of 2nd [q x (q+n+1)] Block

Counting the key marked locations and adding the individual cost required to annihilate the key located elements, we have the following cost equation.

$$\begin{aligned}\text{Cost} = {} & R(i,j,1) + R(i,j,2) + \ldots + R(i,j,k1) + R(i,j,q+n) + R(i,j,q+n-1) + \ldots + R(i,j,k3) \\ & + R(i,j,2) + R(i,j,3) + \ldots + R(i,j,k2) + R(i,j,q+n+1) + R(i,j,q+n) + \ldots + R(i,j,k4)\end{aligned}$$

Where i and j are arbitrary and do not affect the cost.

$$k1 = RD[(q+2)/2], \quad k2 = RD[(q+2)/2],$$
$$k3 = RU[(2q+n+1)/2], \quad k4 = RU[(2q+n+2)/2].$$

Converting this equation using Table 2.5-1, we present the total cost in Table 2.5.4-2. The parameter in Table 2.5-1 is equivalent to q+n+1 for this block.

Table 2.5.4-2. Cost of 2nd [n x (q+n+1)] Block

Operation	Cost
x & ÷	$2c^2 + 2(k1+k2-k3-k4+19/2)c - k1(k1-17/2) - k2(k2-17/2) + k3(k3-21/2) + k4(k4-21/2) + 2$
+ & -	$2c^2 + 2(k1+k2-k3-k4+5)c - k1(k1-4) - k2(k2-4) + k3(k3-6) + k4(k4-6) + 2$
sign	$3(k1+k2-k3-k4+2c)$

where $c = q+n+1$ and k1 to k4 are defined as above.

3) Cost of 3rd [q x (q+n+1)] Block

Counting the key marked locations and adding the individual cost required to annihilate the key located elements, we have the following cost equation.

$$\text{Cost} = R(i,j,1) + R(i,j,2) + \ldots + R(i,j,k5) + R(i,j,2) + R(i,j,3) + \ldots + R(i,j,k6)$$

where i and j are arbitrary and

$$k5 = RD[(q+1)/2], \quad k6 = RD[(q+2)/2].$$

Converting this equation using Table 2.5-1, we present the total cost in Table 2.5.4-3. The parameter in Table 2.5-1 is equivalent to q+n+1 for this block.

Table 2.5.4-3. Cost of 3rd [n x (q+n+1)] Block

Operation	Cost
x & ÷	2(k5+k6-1)c - k5(k5-17/2) - k6(k6-17/2) - 15/2
+ & -	2(K5+k6-1)c - k5(k5-4) - k6(k6-4) - 3
sign	3(k5+k6-1)

where c = q+n+1 and k5 and k6 are defined as above.

Note that the cost for successive blocks (i.e., 4th block, 5th block and so on) as well as the last block which is used only in the case B system, is the same as the cost for the 3rd block obtained in Table 2.5.4-3.

Case B System:

The system matrix for case B is of dimension {[q+n+(M-1)q+q] x [q+n+1]}. The extra block used in addition to the case A system is the far bottom block shown in Figure 2.5.4-1. The cost of this extra block is the same as the cost of the 3rd block found previously. Therefore, no new calculation is necessary other than combining all of their individual costs to obtain a total cost.

Finally, the total cost for an upper triangularization of GTU in a parallel process is obtained as follows:

1) System Case A

Total cost = Cost of [n x (q+n+1)] block + Cost of 2nd [q x (q+n+1)] block
+ (M-2) times the cost of 3rd [q x (q+n+1)] block

2) System Case B

Total Cost = Total Cost of Case A + Cost of 3rd [q x (q+n÷1)] block

Combining all costs according to these equations, the total cost is presented in Table 2.5.4-4.

Table 2.5.4-4. Total Cost of GTU in Parallel Process Block

Operation	Total Cost	
	Case A	Case B
x & ÷	$2n^2 + [4q+(11/2)]n + (15/2)q - 19$	$2n^2 + (4q+11/2)n + (15/2)q - 19$
	$+ 2c^2 + 2(k1+k2-k3-k4+19/2)c$	$+ 2c^2 + 2(k1+k2-k3-k4+19/2)c$
	$- k1[k1-17/2] - k2(k2 -17/2)$	$- k1(k1-17/2) - k2(k2-(17/2)$
	$+ k3(k3-21/2) + k4(k4-21/2) + 2$	$+ k3(k3-21/2) + k4(k4-21/2) + 2$
	$+ 2(M-2)(k5+k6-1)c - (M-2)k5(k5-17/2)$	$+ 2(M-1)(k5+k6-1)c - (M-1)k5(k5-17/2)$
	$- (M-2)k6(k6-17/2) - 15(M-2)/2$	$- (M-1)k6(k6-17/2) - 15(M-1)/2$
+ & -	$2n^2 + (4q+1)n + 3q - 10$	$2n^2 + (4q+1)n + 3q - 10$
	$+ 2c^2 + 2(k1+k2-k3-k4+5)c$	$+ 2c^2 + 2(k1+k2-k3-k4+5)c$
	$- k1(k1-4) - k2(k2-4) + k3(k3-6)$	$- k1(k1-4) - k2(k2-4) + k3(k3-6)$
	$+ k4(k4-6) + 2 + 2c(M-2)(k5+k6-1)$	$+ k4(k4-6) + 2 + 2c(M-1)(k5+k6-1)$
	$- (M-2)k5(k5-4) - (M-2)k6(k6-4) - 3(M-2)$	$- (M-1)k5(k5-4) - (M-1)k6(k6-4) - 3(M-1)$
sign	$3n + 3q - 6 + 3(k1+k2-k3-k4+2c)$	$3n + 3q - 6 + 3(k1+k2-k3-k4+2c)$
	$+ 3(M-2)(k5+k6-1)$	$+ 3(M-1)(k5+k6-1)$

Where c = q+n+1, k1 = RD[(q+2)/2], k2 = RD[(q+3)/2], k3 = RU[(2q+n+1)/2],
k4 = RU[(2q+n+2)/2], k5 = RD[(q+1)/2], k6 = RD[(q+2)/2]

2.5.5 Cost of a Global Measurement Update

I. First Method

Sameh & Kuck's scheme is applied with a modification. A R(i,j,k) is now implemented with a R(i,i-n,k) rather than with R(i,i-1,k). The GMU matrix system of dimension [(M+1)n x (n+1)] is shown in Figure 2.3.5.4-1. Although there are M+1 identical blocks in the GMU system, we take the first three blocks to show the critical locations as defined in section 2.5.4.

$$
\begin{array}{ccccccccc}
 & & & n & & & & 1 & \\
 & x & x & x & x & x & x & x & \\
 & o & x & x & x & x & x & x & \\
 & o & o & x & x & x & x & x & \\
n & o & o & o & x & x & x & x & \text{1st Block} \\
 & o & o & o & o & x & x & x & \\
 & o & o & o & o & o & x & x & \\
 & & & & & & & & \\
 & x\# & x\# & x\# & x\# & x\# & x\# & x & \\
 & o & x\# & x\# & x\# & x\# & x\# & x\# & \\
 & o & o & x & x & x & x & x & \\
n & o & o & o & x & x & x & x & \text{2nd Block} \\
 & o & o & o & o & x & x & x & \\
 & o & o & o & o & o & x & x & \\
 & & & & & & & & \\
 & x\# & x\# & \mathbf{R5} & x & x & x & x & \\
 & 0 & x\# & x\# & \mathbf{R6} & x & x & x & \\
 & 0 & 0 & x & x & x & x & x & \\
n & 0 & 0 & 0 & x & x & x & x & \text{3rd Block} \\
 & 0 & 0 & 0 & 0 & x & x & x & \\
 & 0 & 0 & 0 & 0 & 0 & x & x &
\end{array}
$$

Figure 2.5.5-1. Critical Locations in GMU Sub-blocks

Figure 2.5.5-1 shows the key and critical locations for the 2nd and 3rd blocks. The locations marked with a # sign are key locations and critical locations are labeled **R5** and **R6**. We first find the exact column locations of **R5** and **R6**.
As we did in section 2.5.4, imagine a coordinate system having a horizontal axis c and vertical axis r and superimpose it onto Figure 2.5.5-1 such that the element (or location) of the far bottom left corner coincides with the coordinate (1,1). Then we can set up linear equations for the column locations of **R5** and **R6**.

For **R5**, we have following equations.

$$r = -2(c-1) + (2n-1)$$
$$r = n$$

Solving these two equations simultaneously, we find

$$c = (n+1)/2.$$

Since c must be an integer, we convert it to its closest integer value using a round down function.

$$c = RD[(n+1)/2]$$

The integer c represents the column location of **R5**.
We assign

$$k7 = RD[(n+1)/2].$$

For **R6**, we have following equations:

$$r = -2(c-1) + (2n-1)$$
$$r = n - 1$$

Solving these two equations simultaneously, we find

$$c = (n+2)/2$$

Since c must be an integer, we convert it to its closest integer value using a round down function.

$$c = RD[(n+2)/2]$$

The integer c represents the column location of **R6**.
We assign

$$k8 = RD[(n+2)/2].$$

The cost for a total annihilation of each block is as follows:

1) Cost of 2nd Block

Cost = R(i,j,1) + R(i,j,2) + R(i,j,3) + ... + R(i,j,n)
+ R(i,j,2) + R(i,j,3) + ... + R(i,j,n+1)

Converting these terms to cost terms using Table 2.5-1 and simplifying them, we have the result (noting that the parameter c in Table 2.5-1 is equivalent to (n+1) for this block)

Operation	Cost
x & ÷	n(4c-2n+15)
+ & -	2n(2c-n+3)
sign	6n

2) Cost of 3rd Block

Cost = R(i,j,1) + R(i,j,2) + R(i,j,3) + ... + R(i,j,k7)
+ R(i,j,2) + R(i,j,3) + ... + R(i,j,k8)

where k7 and k8 are defined as above.

Converting these terms to cost terms using Table 2.5-1 and simplifying, we have the result (Noting that the parameter c in Table 2.5-1 is equivalent to (n+1) for this block)

Operation	Cost
x & ÷	(2c-k7+17/2)k7 + (2c-k8+15/2)(k8-1)
+ & -	(2c-k7+4)k7 + (2c-k8+3)(k8-1)
sign	3(k7+k8-1)

The cost for each successive block (4th, 5th, ... , M+1) is equal to the cost for the 3rd block just found.

Therefore, the total cost for an upper triangularization of a GMU in a parallel process with a modified Sameh and Kuck's scheme is as follows.

Total Cost for a GMU = Cost of 2nd Block + (M-1) times the Cost of 3rd Block

Using the cost tables obtained above for the 2nd and 3rd blocks, and adding cost terms according to this equation, we have Table 2.5.5-1 for a total GMU cost.

Table 2.5.5-1. Cost of a GMU using Modified Sameh and Kuck's Scheme

Operation	Cost
x & ÷	(2n+19)n + (M-1)k7(2n-k7+21/2) + (M-1)(k8-1)(2n-k8+19/2)
+ & -	2n(n+5) + (M-1)k7(2n-k7+6) + (M-1)(k8-1)(2n-k8+5)
sign	6n + 3(M-1)(k7+k8-1)

The parameters in Table 2.5.5-1 correspond to those of Figure 2.3.5.4-1 which shows the GMU system matrix structure. The parameters k7 and k8 are defined as,

$$k7 = \mathrm{RD}[(n+1)/2], \qquad k8 = \mathrm{RD}\,[(n+2)/2],$$

where RD stands for the round down function.

II. Second Method

Now, we present another approach for a parallel process (annihilation) of a GMU system. This method takes advantage of the uniqueness of the GMU data structure. A GMU system matrix consists of (M+1) identical standard blocks, [n x (n+1)] in dimension. Blocks are paired together for a parallel annihilation process. One example of pairing blocks is shown below.

$$\begin{array}{cccccccc}
 & \multicolumn{7}{c}{n+1} \\
 & \multicolumn{7}{c}{\text{1st Block}} \\
 & x & x & x & x & x & x & x \\
 & o & x & x & x & x & x & x \\
 & o & o & x & x & x & x & x \\
n & o & o & o & x & x & x & x \\
 & o & o & o & o & x & x & x \\
 & o & o & o & o & o & x & x \\
 & \multicolumn{7}{c}{\text{2nd Block}} \\
 & 1 & 2 & 3 & 4 & 5 & 6 & x \\
 & o & 1 & 2 & 3 & 4 & 5 & 9 \\
 & o & o & 1 & 2 & 3 & 4 & 8 \\
n & o & o & o & 1 & 2 & 3 & 7 \\
 & o & o & o & o & 1 & 2 & 7 \\
 & o & o & o & o & o & 1 & 7
\end{array}
\qquad
\begin{array}{ccccccc}
\multicolumn{7}{c}{n+1} \\
\multicolumn{7}{c}{\text{3rd Block}} \\
x & x & x & x & x & x & x \\
o & x & x & x & x & x & x \\
o & o & x & x & x & x & x \\
o & o & o & x & x & x & x \\
o & o & o & o & x & x & x \\
o & o & o & o & o & x & x \\
\multicolumn{7}{c}{\text{4th Block}} \\
1 & 2 & 3 & 4 & 5 & 6 & 10 \\
o & 1 & 2 & 3 & 4 & 5 & 9 \\
o & o & 1 & 2 & 3 & 4 & 8 \\
o & o & o & 1 & 2 & 3 & 7 \\
o & o & o & o & 1 & 2 & 7 \\
o & o & o & o & o & 1 & 7
\end{array}
\qquad
\begin{array}{ccccccc}
\multicolumn{7}{c}{n+1} \\
\multicolumn{7}{c}{\text{5th Block}} \\
x & x & x & x & x & x & x \\
o & x & x & x & x & x & x \\
o & o & x & x & x & x & x \\
o & o & o & x & x & x & x \\
o & o & o & o & x & x & x \\
o & o & o & o & o & x & x \\
\multicolumn{7}{c}{\text{6th Block}} \\
1 & 2 & 3 & 4 & 5 & 6 & 11 \\
o & 1 & 2 & 3 & 4 & 5 & 9 \\
o & o & 1 & 2 & 3 & 4 & 8 \\
o & o & o & 1 & 2 & 3 & 7 \\
o & o & o & o & 1 & 2 & 7 \\
o & o & o & o & o & 1 & 7
\end{array}$$

and so on.

The integer number in even numbered blocks indicates the step of annihilation between the pair of blocks. This process can be done simultaneously among all even numbered blocks. After half of the (M+1) blocks are processed in this way, we again pair blocks among unprocessed blocks and another simultaneous annihilation is done. This process continues until all blocks have been processed.

Following this idea, we observe that:

1. The number of steps required to annihilate the n x n area, within the standard block, is n.
2. The number of steps required for the last column (dimension = n) is not unique. The number of steps required to annihilate a column of dimension n can be expressed as below.

Column Dimension (n)	Number of Steps Required
$n = 2$	1
$2^0 < n \leq 2^1$	2
$2^1 < n \leq 2^2$	3
$2^2 < n \leq 2^3$	4
$2^3 < n \leq 2^4$	5
.	.
.	.
$2^{p-1} < n < 2^p$	$p+1$

where p is an integer, $p = RU(\log_2 n)$, where RU stands for round up of argument.

3. For (M+1) standard blocks, (M+1)/2 standard blocks can be simultaneously processed in the first round, then (M+1)/4 blocks in the second round, and (M+1)/8 blocks in the third round and so on, provided that $M+1 = 2^q$, where q is an integer. For a general M, let M' = M+1 then we have the following table.

Number of Standard Blocks (M')	Number of Round Required (to exhaust blocks)
$M' = 1$	0
$2^0 < M' \leq 2^1$	1
$2^1 < M' \leq 2^2$	2
$2^2 < M' \leq 2^3$	3
$2^3 < M' \leq 2^4$	4
.	.
.	.
$2^{q-1} < M' < 2^q$	q

where q is an integer, $q = RU(\log_2 M')$. "Round" stands for a unit period required to annihilate a standard block completely.

4. The number of steps required to annihilate the second block is one step (the last step) less than the number required for the other standard block. (The element at the top right corner is not annihilated.)

The step by step cost is listed for a total annihilation of a standard block.

Step	Cost in terms of R(i,j,k)
1	R(3n+1,2n+1,1) ;Expressed based on 4th block.
2	R(3n+1,2n+2,1)
3	R(3n+1,2n+3,3)
.	.
.	.
n	R(3n+1,3n,n)
n+1	R(4n,3n+1,n+1)
.	.
.	.
$n+1+RU(\log_2 n)$	R(3n+1,n+1,n+1)

where RU Stands for a round up function.

Adding the above step by step cost terms and converting the sum to an actual cost using Table 2.5-1, we have the total cost for a standard block in Table 2.5.5-2. The parameter c in Table 2.5-1 represents n+1 in the standard block.

Table 2.5.5-2 Cost for a total annihilation of a standard block

Operation	Count
x & ÷	$(n+21/2)n + (19/2)[RU(\log_2 n)+1]$
+ & -	$(n+6)n + 5[RU(\log_2 n)+1]$
sign	$3[n+1+RU(\log_2 n)]$

The total cost for an upper triangularization of a GMU system will then be the number of round ups (defined above in item 3) times the cost of a standard block. The number of round ups for M+1 blocks was found to be $RU[\log_2 (M+1)]$. Table 2.5.5-3 shows the total cost of GMU system using the second method.

Table 2.5.5-3. Cost of a GMU using 2nd Method

Operation	Count
x & ÷	$RU[\log_2 (M+1)]\{(n+21/2)n + (19/2)[RU(\log_2 n)+1]\} - 19/2$
+ & -	$RU[\log_2 (M+1)]\{(n+6)n + 5[RU(\log_2 n)+1]\} - 5$
sign	$3RU[\log_2 (M+1)][n+1+RU(\log_2 n)] - 3$

2.6 Implementation of Parallel Processing

Parallel processing for speedup of matrix factorization (triangularization) employing the Fast Givens method can be implemented in electronic hardware. As we see from the cost analysis section, a reduction of cost (or speedup) can be obtained in two ways; first, from use of a special hardware processor (cell) which is designed for an optimum plane rotation, and second, from use of multiple processors for parallel processing. We attempt to implement both methods together for best processing performance.

2.6.1 Cell Structure and Operation

The block diagram of a hardware cell is shown in Figure 2.6.1-1. This cell is a hardware version of the Fast Givens algorithm shown in Figure 2.3.4-1. The cell consists of floating point multipliers, dividers, adders, a comparator, multiplexers, sign changers, and vector and shift registers with sequence control. The cell also contains a local memory large enough to hold two identity tokens (t_P and t_Q), two scale factors (d_P and d_Q), a column number (p) which indicates the column location where a zero would be produced, and two rows of vectors ($a_{P,1},...,a_{P,N}$, $a_{Q,1},...,a_{Q,N}$). The plane rotation between these two rows is done completely in this cell, producing two rows of altered elements. The element indicated by the column number P of the second row (Q) vector becomes a zero as a result of the plane rotation. The operation of the cell is briefly stated as follows:

1. The data stream mentioned above is entered into the cell's memory from either a host computer or a neighbor cell.
2. The comparator output signal labeled as "$\gamma > 1$" serves to select the proper mode of data before the actual matrix multiplication is performed by the multiplier and adder arrays in the bottom section of the cell.
3. Two rows of altered vectors, a_P' and a_Q' are generated as a result of the matrix multiplication process.
4. The comparator output signal, $\gamma > 1$, is also used to identify the row information of the newly generated output vectors (a_P', a_Q') and to control the sequence of outgoing data to a neighbor cell or a host computer. For instance, if the signal, $\gamma > 1$, is true (or 1) then the contents of a_P' are interchanged with the contents of a_Q' before being sent out. If the signal, $\gamma > 1$, is not true (or 0), then no action is required.

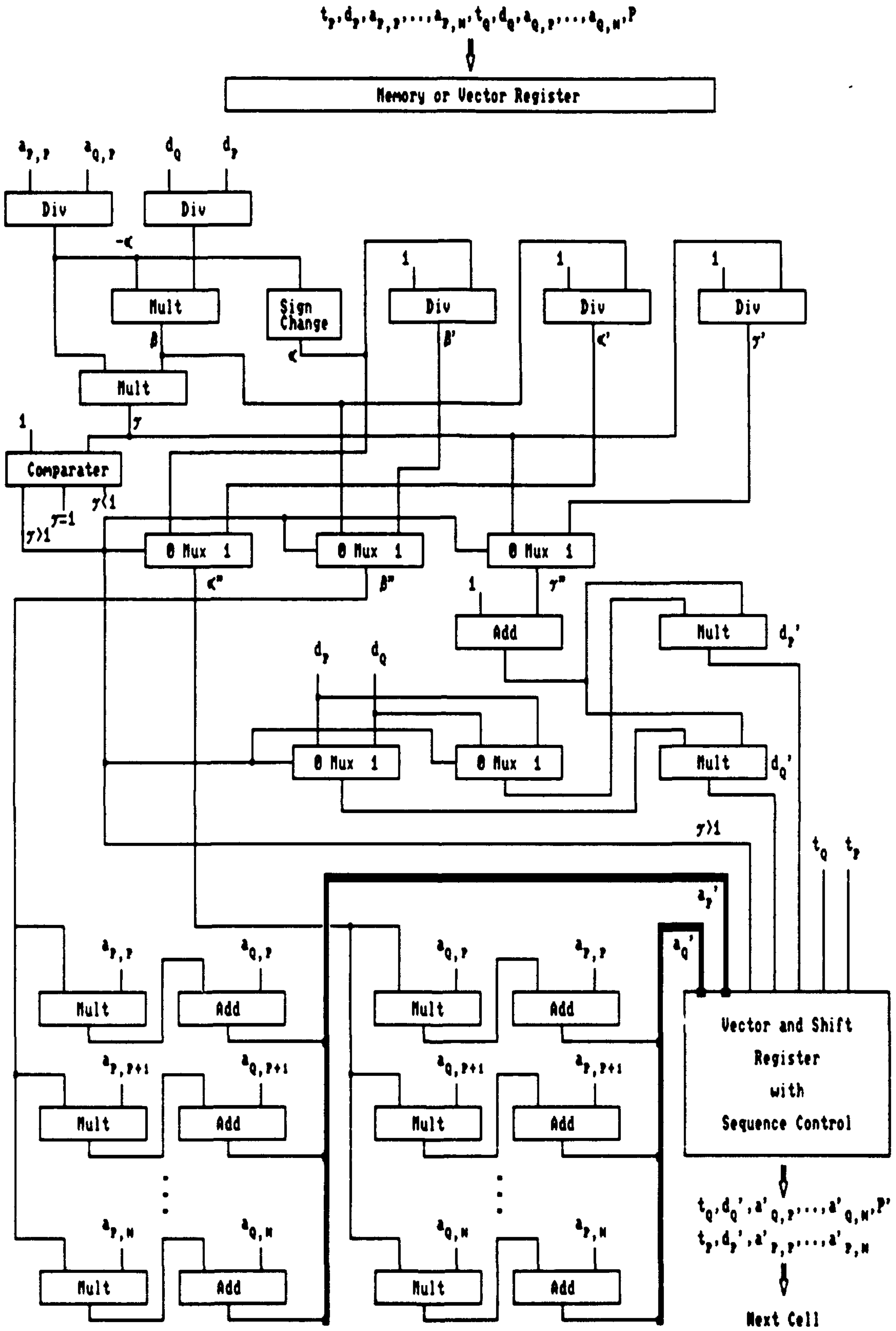

$t_P, d_P, a_{P,P}, \ldots, a_{P,N}, t_Q, d_Q, a_{Q,P}, \ldots, a_{Q,N}, P$
Memory or Vector Register
Div
Mult
Sign Change
Comparater
0 Mux 1
Add
Vector and Shift Register with Sequence Control
$t_Q, d_Q', a'_{Q,P}, \ldots, a'_{Q,N}, P'$
$t_P, d_P', a'_{P,P}, \ldots, a'_{P,N}$
Next Cell

Figure 2.6.1-1. Block diagram of the Hardware Cell

2.6.2 Cost of a Plane Rotation Using a Single Cell

The cost of a plane rotation, R(i,j,k), is shown below in Table 2.6.2-1. The notation, R(i,j,k), represents the Fast Givens Rotation between the ith and jth rows to produce a zero at the (i,k) location. This cost is based on a sequential execution of the algorithm (shown in Table 2.3.4-1) for a single plane rotation. As we see, the cost depends on two parameters, c and k. The parameter c represents the length of the i and j row vectors, and k represents the column location of an element annihilated by the rotation.

Table 2.6.2-1. Cost of a Plane Rotation in Sequential Execution

Operation	Cost
x & ÷	2(c-k) + 19/2
+ & -	2(c-k) + 5
sign	3

Therefore, the cost varies with the column location of the annihilated element. Assuming total annihilation of a vector of length c, the average cost will be,

Operation	Cost
x & ÷	(c-1) + 19/2
+ & -	(c-1) + 5
sign	3

But this cost is drastically reduced (time step wise) by executing the same rotation using the cell we propose in Figure 2.6.1-1. The new cost for an R(i,j,k) rotation will be:

Operation	Cost
x & ÷	5
+ & -	2

Therefore, the new cost for a plane rotation, R(i,j,k), is always a constant and a minimum (7 arithmetic operation cycles). This is due to the internal parallel structure of the cell. Notice that this new cost is even lower that the residual cost required in a sequential execution for a plane rotation. Here, the term "residual" refers to the constant cost terms in the cost table (Table 2.6.2-1).

2.6.3 Architecture of a Parallel System of Cells

The parallel process for a matrix factorization can be implemented using multiple cells simultaneously. Theoretically, m/2 cells are required for the maximum speed of factorization on a matrix of m by n in size. However, it may not be economically feasible when m is so large compared with n, such as when m > 2n. A possible underlying architecture is a SIMD/MIMD machine on the order of n cells which communicates through the shared memory of a host computer.

We propose an implementation of Sameh and Kuck's scheme on a linear array of processing cells. The order of annihilation by the scheme is shown for a matrix of 10 by 8 below. An integer represents the order of time step for an annihilation at which zeroes are created at the respective locations. The time interval of each step is uniform (7 arithmetic operations) due to the internal structure of the cell (all the rotated vector elements are processed simultaneously -- independent of column location).

```
*
9  *
8  10 *
7  9  11 *
6  8  10 12 *
5  7  9  11 13 *
4  6  8  10 12 14 *
3  5  7  9  11 13 15 *
2  4  6  8  10 12 14 16
1  3  5  7  9  11 13 15
```

There is an obvious solution which makes use of an array of n cells, each cell C_k performing all the rotations R(i,i-1,k), $k+1 \leq i \leq n$. The notation, R(i,i-1,k), represents the Givens Rotation between rows i and (i-1) to produce a zero at the (i,k) location. This repartition of the rotations among the cells is represented by Figure 2.6.3-1.

```
*
C1 *
C1 C2 *
C1 C2 C3 *
C1 C2 C3 C4 *
C1 C2 C3 C4 C5 *
C1 C2 C3 C4 C5 C6 *
C1 C2 C3 C4 C5 C6 C7 *
C1 C2 C3 C4 C5 C6 C7 C8
C1 C2 C3 C4 C5 C6 C7 C8
```

Figure 2.6.3-1. Partition of Annihilations with n Processing Cells

The number of cells required can be further reduced. Rather than using n cells per column as shown above, it can be done with less cells if we let the rows of the matrix move backwards as soon as all of the rotations of the first column are completed. The repartition of the rotations among the cells is given in Figure 2.6.3-2.

```
*
C1 *
C1 C1 *
C1 C2 C1 *
C1 C2 C2 C1 *
C1 C2 C3 C2 C1 *
C1 C2 C3 C3 C2 C1 *
C1 C2 C3 C4 C3 C2 C1 *
C1 C2 C3 C4 C4 C3 C2 C1
C1 C2 C3 C4 C5 C4 C3 C2
```

Figure 2.6.3-2. Partition of Annihilations with less than n Processing Cells

This method loses its advantage of less cells being required over the former n-cell method when the number of rows of the matrix grows. Eventually, this method requires n cells just like the former method. *The Table* 2.6.3-1 below shows a comparison among the three methods described.

Table 2.6.3-1. Comparison of Partition Methods

Matrix Size	Number of Cells Required		
	m/2 cell	n cell	Less cell
8 x 8	4	8	4
10 x 8	5	8	5
16 x 8	8	8	8
20 x 8	10	8	8
30 x 8	15	8	8
40 x 8	20	8	8

We propose to implement the last method (less than n) with a linear array of cells as depicted in Figure 2.6.3-3.

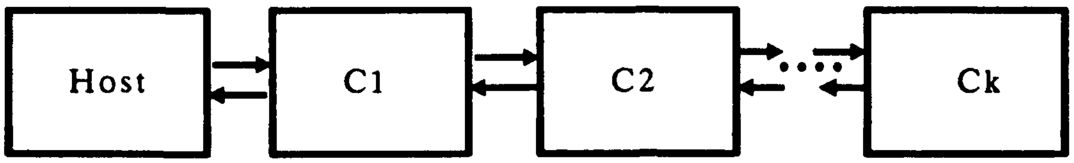
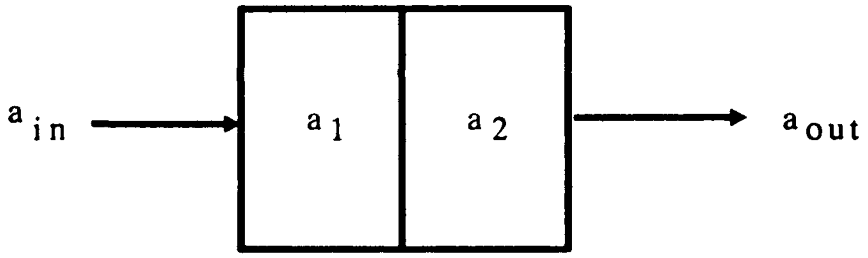


Figure 2.6.3-3. Linear Array of Cells

There are two phases in the operation of a cell. In the beginning phase, at each time step t≤n-1, a row of data moves rightward from the host computer, and each cell C_i operates as indicated in the following.



- Perform a rotation between rows a_1 and a_2 defined as $R(a_2,a_1,k)$, where k is chosen to annihilate the leftmost non-zero element of a_2.
- Send a_2 to the right: $a_{out} \leftarrow a_2$
- Store a_1 and a_{in} in the local memory: $a_2 \leftarrow a_1$, $a_1 \leftarrow a_{in}$

In the last phase, from step t=n up to t=2n-2, the flow of data is reversed and the cells operate in a similar fashion. During this phase, cell C_1 delivers to the host a new row of the resulting matrix A' at each time step.

3. Conclusions

Creating accurate tracks of multiple airborne targets from multiple sensors, in real time, can be a computationally demanding process. Measurements from each sensor must first be correlated with each track. Then, after a correct association is made, the track can be updated to derive a new estimate. A variety of algorithms for performing these processes of "data association" and "track updating" have been described in the literature. Our approach is to perform hypothesis testing based upon the traditional method of Maximum Likelihood, but within a distributed filtering environment. This results in a large reduction in the number of floating point computations required to generate the complete set of likelihood function values.

This final report describes results obtained over a 6 month Phase I project. The primary mathematical operation performed by the distributed filter is matrix triangularization. Thus, this research focused on understanding algorithms for performing this operation, as well as their parallelization.

Three methods based on the orthogonal reduction were reviewed. They are the Householder, Givens, and Fast Givens methods. Gaussian elimination seemed to be an attractive alternative in that it is less costly than those based on orthogonal reduction, but this method is not numerically stable and requires pivoting.

An analysis of computational cost was performed for Householder and Fast Givens methods. Although the Householder method is superior to the Fast Givens method for a generally dense matrix factorization, the Fast Givens method well outperforms the Householder in triangularizing the Local Time Update, Local Measurement Update, and Global Measurement Update matrices of our distributed filter. On the other hand, triangularization of the filter's Global Time Update matrix was more efficiently done using the Householder transformation. This is due to the sparse data structure of the matrix.

The concept of downdating and updating was reviewed along with algorithms from LINPACK. While the updating process is no different from a total annihilation process of the newly added observations (appearing as rows of data), downdating is a backwards process which removes the contribution made by the eliminated observations, from the transformed (triangularized) matrix. The cost of downdating was obtained based on the subroutine "SCHDD" from LINPACK.

The "Sameh and Kuck's" scheme and "Greedy" scheme were reviewed as possibilities for parallelization. Both schemes were modified in order to best suit the block upper-triangular nature of the system matrices. Finally, a hardware processing "cell" which performs a plane rotation using the Fast Givens method, was designed. A linear array of cells following Sameh and Kuck's parallel scheme was proposed.

Although positive results were obtained, it is thought that the funding needed to complete the development of a prototype processor would be prohibitively high under the auspices of this Small Business Innovative Research program. A customized VLSI design approach would probably entail funding at the level of 10 million dollars, whereas Phase II SBIR funding is typically at the level of 500 thousand dollars. Thus, we recommend that an off-the-shelf multi-processor be purchased (or Government furnished) in Phase II, with time better spent in developing efficient code for using it (to perform the triangularization process in parallel to the fullest extent possible).

References

[1] G. Bierman and M. Belzer, "A Decentralized Square Root Information Filter/Smoother", Proceedings of the IEEE Control and Decision Conference, 1985.

[2] M. Belzer and Y. Cho, "Micro-computer Network Architecture for Range Instrumentation Application", MTI final report to the White Sands Missile Range, NM, 1988.

[3] James M. Ortega, Introduction to Parallel and Vector Solution of Linear Systems, Plenum Press, 1988.

[4] William W. Hager, Applied Numerical Linear Algebra, Prentice Hall, 1989.

[5] G H. Golub and C. F. Vanloan, Matrix Computations, The Johns Hopkins University Press, 1983.

[6] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, LINPACK User's Guide, Siam, 1979.

[7] M. Consnard, P. Quinton, Y. Robert, and M. Tchuente, Parallel Algorithms and Architectures, North-Holland, 1986.

Distribution List

Commander
US Army White Sands Missile Range
ATTN: STEWS-ID-T
White Sands Missile Range, NM 88002-5143

Defense Technical Information Center
Cameron Station
Alexandria, VA 22304-614